RetroMagazine

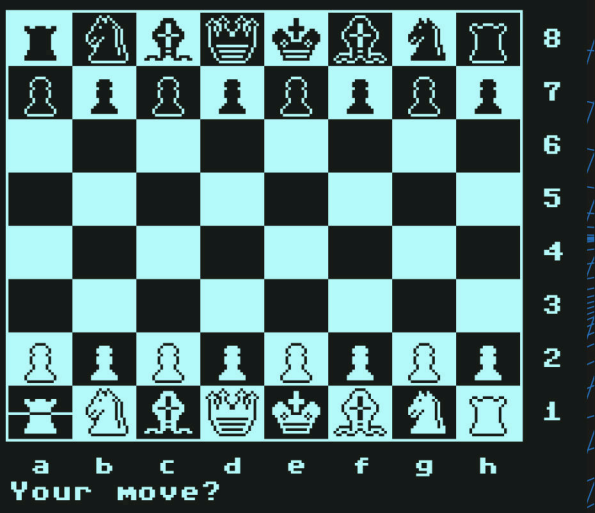
simply retro

Issue 0 Year 1 - May 2020 - http://www.retromagazine.net - Free Publishing

The best version of BASIC

Sinclair QL:
mistakes, misfortune and regrets

Interview with Gideon Zweijtzer
The man behind the U1541 II+
and the Ultimate 64



Cyrus VS Colossus:
which was the best chess program of the 80s?

The Pawn:
a classic milestone in interactive fiction

Don't ever buy a Vectrex!

The D64 file format - part 1

Civilization: the Sid Meier masterpiece

Special issue entirely
in English

EDITORIAL

Welcome to our first international issue!

RetroMagazine is a well underway project started in October 2017 by a small group of Italian retrocomputing enthusiasts.

Yes, we know, there are many magazines dedicated to retrogames and they are gorgeous; so why the need for an additional homebrew fanzine?
How many times reading those publications have you felt that something was missing? There are lot of nice pictures and a good variety of games, but... where is the code? Where are the explanations of programming techniques? Where is the real experience of the end users?

The idea behind our project is to reproduce the same feeling as the glorious magazines back in the day. Magazines like Amstrad Computer Users, Bit, Compute!'s Gazette, Input... They all taught tons of pimply boys the basis of coding on their home computers! Those magazines used to contain a good balance of programming examples, hardware insights and game reviews.

In memory of those magazines we adopted the shape of a PDF fanzine, instead of a more modern blog or website, to fully revitalize the spirit of the good old times.
Do you remember the feeling while awaiting for the next issue? We want to recreate that and also give a second chance to anybody who missed out on learning these things as a kid.

We created a virtual editorial board to give shape to our dreams. Every decision concerning the life of RetroMagazine is discussed there, where everybody's opinion counts because we are driven by the same passion. We have already released 22 issues in Italian and they have been well received by the Italian retro enthusiast communities.

Now it's time for another big step for RetroMagazine. As I said at the beginning of this editorial, this is our first international issue. Being a homebrew project we could not really afford for professional translators and we don't have the time to translate our works ourselves; this is a hobby, we also have jobs and families! This is the reason why it took so long before publishing the first issue completely in English.

The number you virtually hold in your hands is a pilot edition. As I wrote in my first Italian editorial, we need you. If you like our work, then let us know and if you want to contribute, well, the door is open!

So, have a nice read of this special zeroth issue!
I hope you'll find it interesting and fun as we did when working on it.

Francesco Fiorentini

SUMMARY

People involved in the preparation of this issue

- Robin Jubber
- Francesco Fiorentini
- David La Monaca
- Alberto Apostolo
- Gianluca Girelli
- Leonardo Giordani
- Giorgio Balestrieri

More credits

- Graphic support: Irene G. Valeri
- Cover: Flavio Soldani
- Page setting: Francesco Fiorentini, Marco Pistorio
- Proof-reading: Francesco Fiorentini, David La Monaca, Robin Jubber, Giorgio Balestrieri, Alberto Apostolo

DISCLAIMER ABOUT THE ENGLISH VERSION OF RETROMAGAZINE

SOME OF THE FEATURED ARTICLES/CONTENTS WERE ORIGINALLY WRITTEN IN ENGLISH BY THEIR RESPECTIVE AUTHORS. THE OTHER ARTICLES HAVE BEEN AUTOMATICALLY PARSED BY MODERN TRANSLATION ENGINES AND THEN REVIEWED AND PROOF-READ BY MEMBERS OF THE EDITORIAL STAFF.

The best version of BASIC

by Robin Jubber

For a great many games coders, especially the older generation of this young industry, the BBC User Guide, explaining the BBC BASIC language, was our Bible. I have two or three original User Guides in my house, just in case I lose my primary copy. They're all falling to pieces after thirty years of service.

I'm a games programmer in my 40s, and like almost every games programmer in their 40s, my first introduction to programming began with BASIC. Every 8-bit micro from the mid-70s onwards had a variation of this influential language built into the hardware, with varying degrees of implementation success. Since those days I've been lucky enough to program in C and C#, unlucky enough to program in languages like C++ and PHP and cursed by the gods themselves to dabble in the horror shows that are Objective-C and Lisp. And those aren't even the worst languages out there. Some of the languages I've coded in no longer exist, or were exclusive to one company or even one game. Others, like C, have declined a little in popularity over the years, but still influence all the new languages, by lending familiar syntax to platforms like Java and C#. But long before I encountered pointers, classes and memory management, I, like all my contemporaries, started coding in Beginners All-purpose Symbolic Instruction Code, a language first invented more than half a century ago.

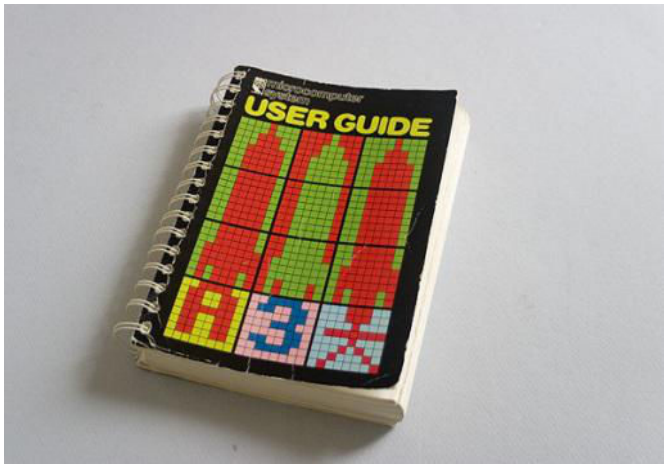
Microsoft cornered the BASIC market for a long time, with the version that Bill Gates and Paul Allen created for the Altair in around 1975. This 8K language rom ended up in many flavours of machine back in the 70s and 80s, including Apple, Commodore, IBM, Tandy, Atari and CP/M computers. Despite its primitive nature, the implementation was a safe bet for garage companies building kit computers and even established hardware manufacturers turned to Microsoft. MS Basic was in essence the bedrock for the entire Microsoft company we know today.

Not every computer manufacturer subscribed to the MS monopoly of course. My first exposure to any form of programming was struggling to construct simple Basic programs for the Sinclair Spectrum at school. Sinclair Basic was a pretty poor version of the language, due in no small part to being limited by the Spectrum's one key input system and unsophisticated editing environment.

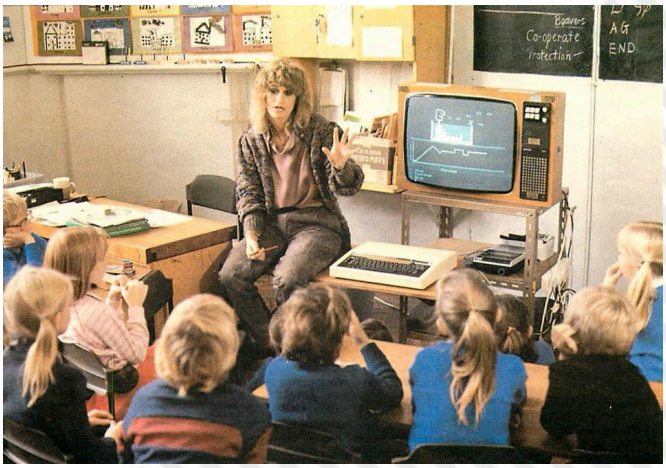
The BBC Micro home computer

The Spectrum's hateful rubber keys were also a profoundly limiting factor. Try typing in a Spectum program using the original keyboard – it's shockingly difficult. Other manufacturers were also experimenting with their own versions of the language, for instance Atari Basic for the XL, Wozniak's Basic for the Apple II and Tiny Basic, which somehow fitted a functional version of the language into just a couple of kilobytes.

Luckily for me my first personal computer was a BBC Micro – a truly remarkable machine that dominated the UK education market but saw only limited success outside these islands. I have to assume my dad sold a kidney to



The official BBC User Guide



A typical UK primary classroom of the 80s

Sophie Wilson – designer of BBC Micro and ARM chip

pay for the machine. That was fine by me as long as I got to play Elite and shoot some Thargoids.

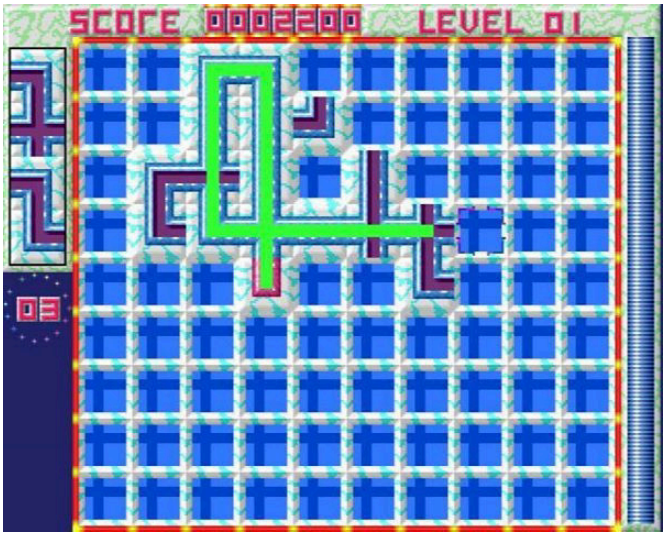
The BBC Computer Literacy project was designed by the UK's public broadcasting corporation to introduce both children and adults to the brand new world of home computing that was opening up at the start of the 1980s. Part of the literacy project involved finding a computer manufacturer who could build an all-purpose machine that would spearhead the television programs associated with the project. A number of manufacturers, primarily in the UK, raced to build a machine that would become the official BBC microcomputer. Acorn Computers, which had started a few years earlier in Cambridge, would become the eventual winner, due in no small part to the powerful and comprehensive version of BASIC incorporated into the machine. The BBC wanted a language that could do everything, would be well structured, and allow users to access the huge array of specialised input and output systems on the machine.

The language was designed and implemented by Sophie Wilson (Roger Wilson at the time) - a British computer scientist who would go on to have an unparalleled career. Sophie Wilson is a name to remember – she was one half of the team that went on to design the ARM RISC processor, which initially sold as a coprocessor module for the BBC Micro. The ARM chip can be considered a pretty impressive legacy of the BBC Micro, with more than 100 billion manufactured. There are probably two or three right next to you now, hidden in every conceivable electronic device on your desk. One of the stated aims for the ARM chip was to run BBC BASIC at the same speed the original 8-bit micro could run assembler. It's fair to say Sophie Wilson knocked that objective out of the park. The ARM was fast. BBC Basic V, the version she made for the Archimedes range of machines, would entirely fit inside the cache of later ARM chips ensuring main ram was rarely accessed by the interpreter. This of course made a very fast language even quicker and was one of the reasons so many multitasking desktop apps on the Archimedes could easily be written in BASIC instead of C or assembler.

I actually had a couple of commercial games on the Arc released that were almost entirely created in BBC Basic V back in the day. The games themselves may not have set the world on fire, but to be able to write full screen games using an interpreted language wasn't something you could do on any other machine.
Sophie Wilson based her version of BASIC on Atom Basic, from an early Acorn machine, along with key changes brought in from languages like COMAL and Pascal. These included proper structural programming features – most notably functions, procedures, repeat loops, if-then-else construction and an assembler built directly into the language. These features were remarkably forward thinking at the time and certainly ensured that young programmers lucky enough to have access to an Acorn machine were well served when they started computing courses and needed to understand proper structure. It may help explain why the UK had such a presence in the early computer games industry as BBC Micros were available in pretty much every school in the country.

On other machines you couldn't really write complex BASIC code without using GOTO and GOSUB commands. These were hard coded jumps in the codebase that forced execution to move to a new area of code. However they made the code inflexible and hard to maintain or understand. GOTO does have its place in programming, especially in error trapping, and isn't quite the code villain it has traditionally been painted, but it's a very low-level command. GOTO essentially acts like a branch instruction in assembler, forcing the processor to jump to a new memory location. The big problem is that just by looking at a line like 10 GOTO 90 you're none the wiser about what you might expect to find at line 90, which would in turn have to jump elsewhere with another GOTO. This leads to spaghetti code and a coding style that doesn't extend well to larger projects. With BBC Basic, the GOTO and GOSUB commands were entirely unnecessary. You could instead write PROC_Draw_Square or FN_Root(10), which makes the code far easier to parse and modify.

I still use BBC Basic today, especially when I'm writing a side project for an older machine like the Vectrex. I can have a BBC emulator running on the desktop to use as a scratch pad for simple coding experiments or generating look up tables in a format the Vectrex will understand. I have even used BBC Basic to generate table data for



A screenshot from GLoop (Archimedes)

JSBeeb – an amazing online BBC emulator

games running on much more powerful hardware, such as the PSP – it's just a handy tool to have lying around.

The editing environment is as powerful and easy to use as I remember from the 1980s and I would strongly recommend fans of 8-bit machines load up JSBeeb (written by a friend of mine called Matt Godbolt, another coder who got started on BBC Basic) to get a taste of what programming in this very immediate environment felt like. Not only could you copy sections of the screen into new lines, you also had trace functionality, sophisticated line renumbering tools, the aforementioned assembler that could be called from within BASIC (and use BASIC functions and constants) and complete control of the BBC Micro graphics capabilities. Other versions of Basic had no graphics or sound commands, with PEEK and POKE required to drop values into individual areas of memory and trigger chip functionality. It is fair to say that BASIC on the C64 was barely useable, which meant many aspiring young coders of the era were denied access to the Commodore machine's impressive graphics and sound chips.

BBC Basic has had a storied history, ending up on Amstrad, Sinclair, Windows, Archimedes and Texas Instruments platforms to name a few. It's even still doing actual work, when almost every other version of BASIC (with the honourable exception of Visual Basic) has long since disappeared. A couple of years ago my brother, who ran a sweet shop at the time, was having trouble with his Point of Sale machine. I took the computer apart, loaded up the code that was crashing, and discovered to my amazement it was running a modern version of BBC Basic.

Elite - the BBC Micro's most famous game

Needless to say that made it particularly easy to fix. It also means that somebody out in the business world is still getting away with writing commercial software using a language he first learnt as a child. That unknown coder is my kind of hero.

External links

• Acorn Computers
https://en.wikipedia.org/wiki/Acorn_Computers

• BBC Basic
http://www.bbcbasic.org

• Microsoft Basic
https://en.wikipedia.org/wiki/Microsoft_BASIC

• Tiny Basic
https://en.wikipedia.org/wiki/Tiny_BASIC

• Elite per BBC Micro
https://en.wikipedia.org/wiki/Elite_(video_game)

• Sophie Wilson
https://en.wikipedia.org/wiki/Sophie_Wilson

• JSBeeb: online Acorn BBC emulator
https://bbc.godbolt.org

• Antigrav, one of Robin's games for Archimedes
http://www.apdl.org.uk/apdlpd/library/files/g/g131.zip

• Fred The Needle, another Robin's game fully written with BBC/Electron's BASIC
https://jubberbbcmicro.webnode.com/fred-theneedle/

• Chip ARM's history - from Acorn to Apple
https://www.telegraph.co.uk/finance/newsbysector/epic/arm/8243162/History-of-ARM-from-Acornto-Apple.html

Don't ever buy a Vectrex!

by Robin Jubber

Don't buy a Vectrex. Sure, without one your console collection is just a bunch of plastic landfill, and yes, it's the greatest console of all time; but still. Don't buy a Vectrex. I bought one and now I have three. This vital information has been withheld from the girlfriend, and because her Italian is as bad as mine, she may never find out, god willing. I also ended up with a stack of expensive homebrew, a ton of wonky peripherals and I recently wrote a game – well, 14 games – for the machine and now my house is full of packaging, soldering irons and plastic carts. I've become a home based manufacturing company – I should have bought a Neo Geo like a sane person.

If you're unfamiliar with the Vectrex, don't feel bad. Certainly it means you're not a proper console collector, know nothing about retro gaming and shouldn't be reading this magazine but on the other hand, your bank balance probably isn't a giant smoking hole in the ground. The Vectrex is a rare and unusual beast. It's the only non-portable with an integrated screen. It's the only console that uses vectors instead of pixels. It's one of a few select consoles where a second controller is almost as expensive as the console itself (that's why I bought the third machine, honey). It has colour graphics, but only in the weirdest possible sense. It was the first console with a 3D headset. Every one comes with slightly wonky graphics, but in a manner unique to that machine. It makes a weird buzzing noise even with the volume at zero. Opening it up for repairs can literally kill you. It has a carrying handle of sorts, presumably so you could take it very, very carefully to your friend's house but the carrying handle is incomplete and sort of sloped, so dropping it is a strong and expensive possibility. It was only in production for about a year and a half. It has around 800 bytes of ram. It is, without doubt, the greatest console of all time.

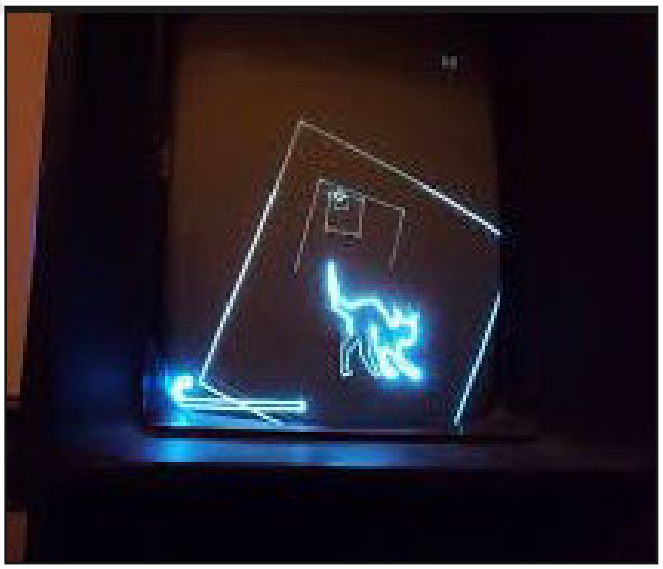
The first thing you'll notice is the screen. Essentially an old black and white CRT on its side, the vector display is like nothing else out there. The simple line-based graphics pop out from the screen in a way that cannot be replicated by an emulator. Your 8 gig graphics card cannot make graphics like this. Older readers may have played the original vector Star Wars in the arcades – this is essentially the same technology, squeezed into something that sits on your desk. Because the graphics are so unusual, and don't use pixels at all, my second mistake was to try writing a little code. Perhaps I could get a simple triangle up on screen. That could be fun. Then I'd get back to writing proper games on machines with millions of kilobytes of ram instead of not quite 1. A little searching on the internet turned up a few snippets of information, such as Christopher Tumber's 1998 text file that helps with some aspects of the machine and 6809 assembler. You'll also need as09, for turning code into binary, and ParaJVE, a not entirely accurate or finished Vectrex emulator. It's also possible to write your code to eeprom and test it that way, but you'll die of old age before you get Hello World world up and running. There is also, incredibly, a full development environment created by a lovely chap called Malban, but I didn't want to bother with that – this was just meant to be a quick experiment after all. But we've already established I may be an idiot.

The 6809 that sits at the heart of the machine is a wonderful cpu. It really only turns up in embedded systems, the Vectrex, the Dragon 32 and the TRS-80. And that's a shame. The architecture is big endian, which means a lot of 16-bit maths is easier for humans to understand. Little endian, which turns up in rival 8-bit cpus, is more convenient for simple logic units to handle a byte at a time, which keeps manufacturing costs down, but doesn't make for such readable or intuitive code.

For a supposedly 8-bit cpu, the 6809 is also very 16-bit capable. It has two 8-bit registers that combine to create a 16-bit word, along with a host of other 16-bit registers and instructions. Most of the code you write will be 8-bit, the usual business of comparing small values and looping over small data structures – but when you need to write some fixed point maths, the 6809 really comes into its own.

For my first experiment, a simple Pong clone, this wasn't all that important. By the third game, a simple Spacewar clone, it had become vital to making smooth movement.

The original Vectrex game console



Drawing shapes with a Vectrex

Essentially you don't want to be restricted to integer mathematics – you can then only move discrete numbers of units per frame – and as the Vectrex only understands 256 units vertically and horizontally, that would mean your minimum movement rate would cross the screen in just a few seconds. So you use 16-bit values, and treat the first byte as the actual coordinate, and the second byte as the fractional part. Very easy with a 6809.

The problem with writing a simple Pong clone is that I still had about 25K of rom space free. Obviously I could have just released the binary file and got on with the day job, but by now I was hooked on understanding what the Vectrex could do and learning a new language. Plus Joanna, my baby daughter, had just turned up out of nowhere so I was essentially on self-imposed paternity leave. So I wrote another game. And another. Then I shuffled memory around a bit, noticed some routines were being used all over the place and found some more spare rom space. So I wrote another game.

At this point the original 32K rom, which is all the Vectrex can address, was getting close to the limit, so my four or five simple games would end up on the internet and I could move on, having scratched a programming itch. I'd also mentioned my Vectrex adventures to some other ancient programmers who've been making games since the dark ages and they were very jealous. No compile times. No producer leaning over my shoulder. No gently explaining to simple minded artists why their artwork is all broken and useless. Just raw assembler, written directly to the metal – proper game coding like we used to do when men still lived in caves.

At this point things took a turn for the worse. I discovered that some clever coder had figured out how to do bank switching on the Vectrex. You could have 64K of code and data, provided you were very careful about how it was initially laid out. If the first chunk of instructions are the same for both banks, you could send a signal to a magical place in the Vectrex hardware, and switch to bank 2. Or back again. Using the same technique, you could also write to a 32 character eprom, in effect giving the Vectrex save and load capabilities. Well, obviously I had to have some of that. Another 32K to play with? This could become a compendium of two player games.

Two player games have a couple of distinct advantages to the lazy incompetent coder. Firstly, the Vectrex simply doesn't have much in the way of two player gaming options, despite two joypad ports. Many games can be played with alternating players, but if you want to show off your fancy new Vectrex to a baffled friend, you need proper simultaneous two player.

The other advantage is that I wouldn't need to write much in the way of artificial intelligence. And to think, my computer science teacher worried I didn't have the right mindset to be a programmer. What a fool! Sure, if I had done things his way I'd be writing database software for a bank and driving around in a Lamborghini, instead of a 13 year old BMW that likes to shut down in the middle of the road for no apparent reason while everybody points and laughs. But I wouldn't get to make spaceships move around on screen, so it's a fair compromise. I think.

By now I was up to 6 or 7 games, including my first properly complicated bit of code, for a single screen worms-style artillery game. Next came a basic stock car game, played from overhead and then Tron. The Vectrex is all about lines so Tron seemed a natural fit. In theory. The unusual thing about the Vectrex is that it has no persistence. Nothing you draw stays on screen for more than a frame – everything has to be redrawn 50 times a second, completely. Nothing else works like this, although modern 3D games essentially have to address the same issue. You have 30000 cpu cycles to get everything drawn and

The Vectrex in action!

The Vectrex 3D Imager helmet

then the entire process has to start again as the previous frame's lines fade quickly from view.

The Vectrex draws graphics in the style of an oscilloscope – the electron beam leaves a trail but it doesn't persist for long. The scaling of the lines you draw also introduces delays and text is especially hard. The OS routines for displaying text are not fast, and grow increasingly distorted the more letters you display. Your 30,000 ticks are constantly under pressure. With Tron I couldn't write the game in the usual manner, by adding a pixel to the head of the snake, and simply deleting a pixel from the tail. Instead I had to store all the lines, including the special case of lines that are longer than 128 units (the maximum on the Vectrex) and also keep extending the line forwards as the player moves around the screen.

This set of lines then had to be packaged up to the OS as a complete vector image, so it could be drawn instantly, for both players. Well, no problem – I have a few hundred bytes free, if I wipe over ram that is used by other games but which Tron doesn't need. That meant the missile system, the explosion system, the block based collision system, the particle system – they can all be repurposed. I just need a collision system that handles line intercepts and a giant queue to handle the line ends. Also two of everything because there are two players. Oh my god that is some ugly code – but it works.

But still – all that rom code left to fill. And I had better fill it fast or my daughter will have left for university by the time this cart is done. So I took the Spacewar code, added gravity, and built a series of caverns for the two ships to navigate. A co-op game on the Vectrex! With 100 rooms! Those 100 rooms nearly killed me – level design is soooo tedious, but eventually it was up and running.

I won't bore you with the critical bugs, the realisation that saving and loading also force a bank switch, the easter eggs I added which nearly broke everything and the amount of data shuffling needed to fit 5 translated languages into the cart. I also had Malban look over the code - "it's very good for a first game, but you probably shouldn't have written something this large and here are all the things wrong with it" - where he went through 22 thousand lines of assembler looking for places where I incremented a loop instead of decrementing it (saving, 1 or 2 bytes) or loaded two registers separately instead of one (saving 1 byte). The man's a mad genius and I'm not ignoring help like that – it all adds up. Bytes and clock cycles are precious when you're coding in the past. Malban also helped me understand the relationship between the scale of vectors and their drawtime, which was invaluable for getting the menu system to run without dropping frames.

So the game is done, except for the hidden Black Vector screensaver which I wrote at the last minute, and the hidden message to my girlfriend (because every girl dreams of getting a feeble apology delivered on antique hardware) and the other hidden features I add because I can see some free bytes just lying around doing nothing. Now I'm up to 14 games, 6 screensavers, 10 utilities, 6 game settings, 5 languages, 11 alternative versions for the main games, 3 bits of hidden seasonal DLC and I get that twitchy feeling when the project is finished and you want to write some more code but there's nothing to write and nowhere to put it anyway!
And that's just the start of my problems. Right now I'm a manufacturing hub because to make a Vectrex game properly you need to create carts, burn roms, solder components onto pcbs, you need to find somewhere that can make boxes, you need to make some artwork, write manuals and their translations (many thanks to all who helped with that), draw posters for that proper 80s feel, edit a web page and that's before you even get to packaging everything up and talking to all your customers.

I also had to find a friendly plastics manufacturer and carefully explain what a Vectrex is, and why I need a colourful two layer piece of polycarbonate with a pretty pattern on it. My first step towards all this was buying a 3D printer in order to make carts. Test cart 1 looked like that bit in Judgement Day where the T-1000 is trying to reform after Linda Hamilton blasts him with a shotgun. My next attempt resembled the special effects from the sci-fi classic The Thing. I'm a coder – I don't know anything about plastics! Now I buy my carts from America where it merely costs all the money.
So hopefully this will give you a slight idea why you shouldn't buy a Vectrex.
I'm serious. You have no idea what you're getting yourself into. Even if it is, without doubt, the very greatest console ever made.

Vectrex Technical Addendum

Here are the minimal steps for Hello World.
1) Download as09.exe from http://www.kingswood-

consulting.co.uk/assemblers/ or try VIDE by Malban, which has an assembler built in.

2) Choose an IDE. I initially went with Visual Studio and created a rule in Tools -> External Tools. Add a new external tool and call it Vectrex Assembler. For Command enter C:\Wherever\as09.exe and for Arguments use –q $(ItemFilename). Tick Use Output Window and leave the other options blank. You will also want to add a keyboard shortcut, usually found in Options, to execute this external tool on the current file. I use F6, but your tastes may vary. Of course you can skip all this and just use VIDE, which has buttons to compile assembler into 6809.

3) Run your code. As09 will output a bin (binary executable) file which you can load with ParaJVE, a Vectrex emulator. ParaJVE can also handle bank switching, but not save/load – although these are features most Vectrex games do not use or need. Again, VIDE has an emulator inside the IDE, and it's more accurate for spotting framerate and draw problems.

4) Write some code. Each Vectrex game has a header, which will be displayed by the OS on bootup. After that header is where your code begins. Here's an example.

5) Become addicted to 6809 assembler and watch your social life disintegrate.

Some handy hints for Vectrex development I wish I knew beforehand.

• The OS routines for printing strings will fail with no explanation if your string is just one character long. You won't have a clue why.

• The longer a string, the more distorted it will become.

• In many cases it is better to scroll a string than display it as one page of text.

• When designing graphics, for instance in Blender, try and make your vector sprite as large as possible – generally to fill the screen – then scale it down to display it. The smaller the scale of a vector list, the less cycles it will take to display. There are subtle aspects to this guidance, but in essence this is a good rule of thumb.

• ParaJVE draws as if running on a magically perfect Vectrex. No such beast exists in the wild.

• Any attempt to save or load will cause a bank switch. Have all your eeprom, initialisation and bank switching code in identical locations in their respective banks.

• Don't worry too much about writing super efficient assembler, except inside critical loops. Vector redraw is going to be your framerate killer, unless you're doing a lot of 3D maths. In modern parlance, you're most likely to be GPU bound.

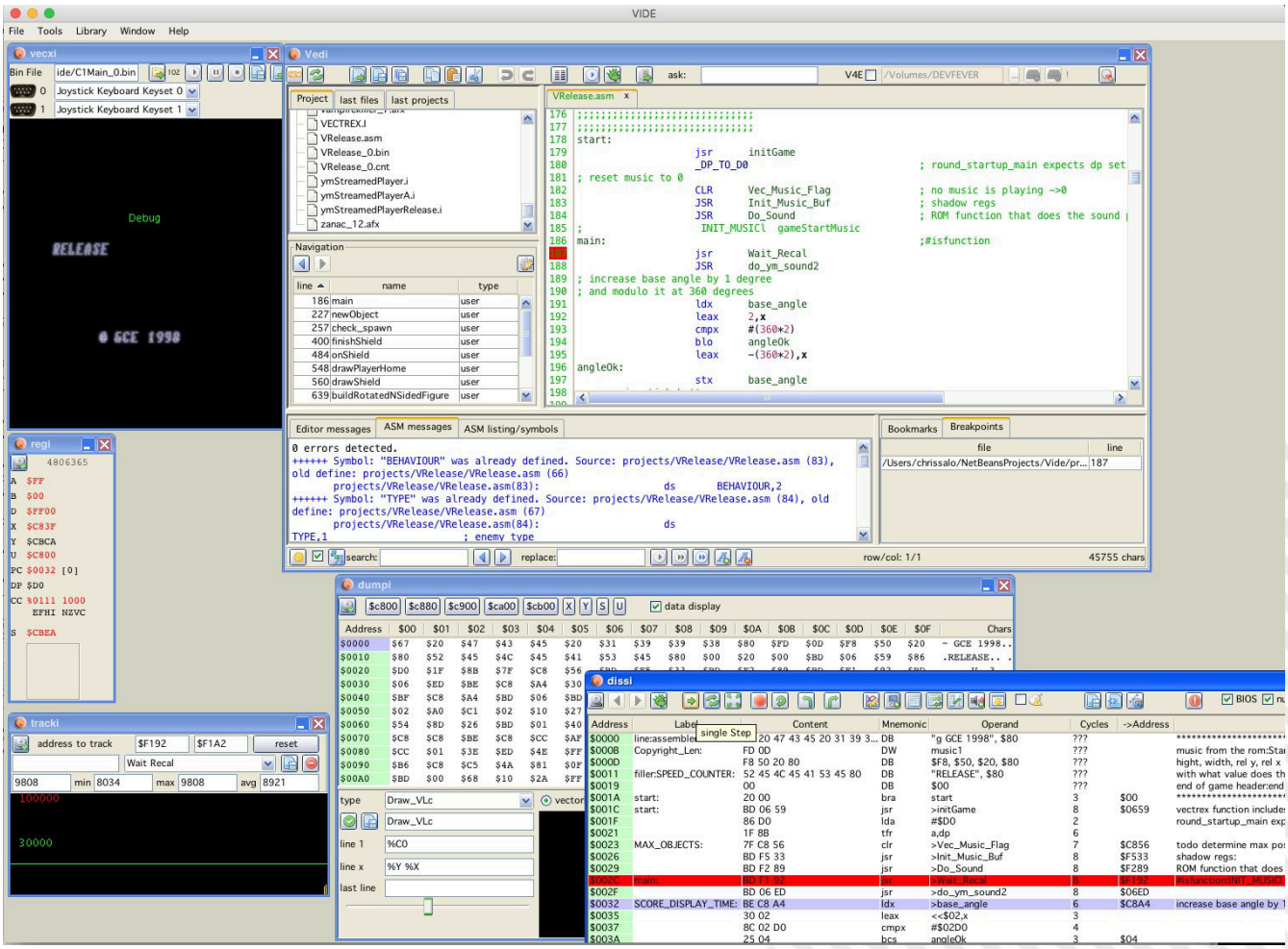
• On the 6809, most instructions, for instance loads, will also affect the status flags.

• This page has a summary of the OS commands and the registers destroyed after jumping into these subroutines http://www.playvectrex.com/designit/chrissalo/bios.htm - which is why a and b will no longer hold their values after moving the beam.

• When using the joysticks, you have to turn their respective features 'on' – for instance movement in x and y, and analogue control. If you don't need these facilities, leave them off and you will save a bit of cpu time.

• Before displaying text at the start of the frame, perform a move, otherwise the text will not show up.

• Going back to 6502 or Z80 will *hurt* after coding this chip.



VIDE - Vectrex Integrated Development Environment

```
; Define some OS routines we will need
OS_Intensity_7F   equ   $F2A9 ; sets the intensity of drawing to maximum
OS_Intensity_A    equ   $F2AB ; this version uses the a register
OS_Wait_Recal     equ   $F192 ; Vectrex BIOS recalibration
OS_Print_Str_D    equ   $F37A ; BIOS print routine
OS_Reset0ref      equ   $F354 ; Call frequently to avoid wobble
OS_Move           equ   $F2FC ; move to location specified by regs a and b
OS_Draw           equ   $F3DF ; draw by amount in registers a and b

; some assembler directives (optimise and start of code in memory)
      opt
      org   0

;*************************************************************************
; HEADER SECTION
;*************************************************************************
      db    "g GCE 2018", $80
dw    $FF8F                   ; address of a (BIOS ROM) tune
db    $FC, $30, $20, -$58     ; height (negative), width, rel y, rel x
db    "JUBBERNAUT", $80       ; game title. db sets aside some bytes in ram
db    0                       ; end of game header

;*************************************************************************
; GAME
;*************************************************************************

Main
jsr   OS_Wait_Recal           ; start of draw cycle
jsr   OS_Intensity_7F         ; set the intensity of the beam - 7F is max


Draw_A_Line
      jsr   OS_Reset0ref      ; Reset the beam
lda   #30
ldb   #-64
      jsr   OS_Move           ; Move to -64,30 (a and b are y and x)
      lda   #0
      ldb   #125
      jsr   OS_Draw           ; draw a horizontal line (0 units in y, 125 in x)


Write_Some_Text
jsr   OS_Reset0ref
      lda   #0                ; OS_Draw corrupts registers a and b
      ldb   #-60
      ldu   #MESSAGE          ; load the 16-bit U register with the string
jsr   OS_Print_Str_D          ; call the OS print routine


Draw_Another_Line
jsr   OS_Reset0ref
      lda   #-30
      ldb   #-64
      jsr   OS_Move
      lda   #0
      ldb   #127
      jsr   OS_Draw

Main_End
      bra   Main              ; and return to the start of the game


MESSAGE     db    "ROBIN IS SKILL!",$80

                              ; the Vectrex font is capitals only.
                              ; Some punctuation and special symbols
                              ; are also supported.
```

Listing 1 - 'Hello World' (...or 'Robin is skill!') in assembly Motorola 6809

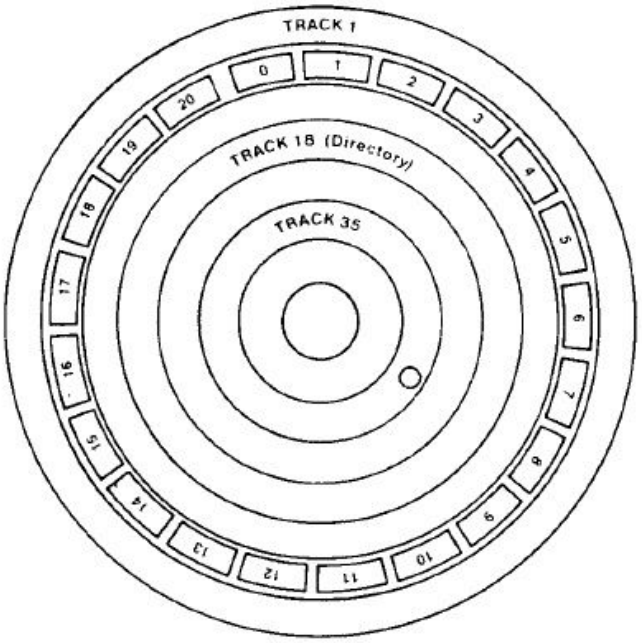
The .D64 format - part 1

by Francesco Fiorentini

If you have ever used a Commodore 64 emulator, you may be familiar with the D64 format. Using a D64 file in an emulator is quite easy: pick up the file, load it into the emulator and then use it as a normal C64 floppy disk in a real 1541 Disk Drive. The emulator will take care of the file and treat it as a real 'physical' C64 floppy disk.

Nothing really fancy so far, we are all used to work with emulators... but if you are still reading this article, it means you want to know more about the structure of a D64 file and how it works. Additionally, since a D64 is the physical representation of a 1541's single-sided disk in a file format, knowing its structure you can easily understand the functionality of a real Commodore 1541 Disk Drive. So let's start our journey.

During the formatting process, the 1541 DOS (Disk Operating System) organizes the disk surface in tracks and sectors. The sectors will be used to store information. The formatting creates a total of 35 concentric tracks, starting from 1 (the outermost one) up to 35 (the innermost one), then each track is divided into a variable number of sectors. Why is there a difference in the number of sectors per track? Does this mean the disk rotates with a different speed based on the header's position? Not really, the disk always rotates at a constant speed of 300 rpm (round per minutes), the difference is provided by varying the clock rate at which data are written on (or read from) the disk. Every disk is divided in 4 areas with a different clock rate.



1. Graphical representation of a 1541 disk (image from the book 'Inside Commodore DOS')

Without overcomplicating our article with tedious math calculations, the 4 different clock rates do generate 4 different areas containing a number of sectors per track (see *Table 1*).

Area	Track	Sectors	Clock rate	Sectors per Area
1	01 - 17	21	307,692 bits/sec	357
2	18 - 24	19	285,714 bits/sec	133
3	25 - 30	18	266,667 bits/sec	108
4	31 - 35	17	250,000 bits/sec	85

Table 1: Areas, tracks, sectors and clock rate

Doing a quick sum of all the sectors of each area we get to know that the total number of sectors in a disk is 683. Every sector is 256 bytes long and it contains, other than the stored data, a header to identify the sector itself.
By multiplying 683*256 we get the value of 174848 bytes which is theoretically the total storage capacity of a diskette formatted with a 1541. By dividing 174848 by 1028 (1k) we get 171KB which is exactly the size of a D64 file as shown by the host file system. But, wait a minute... Why did we use the word 'theoretically'? Because the 1541 DOS reserves some space for itself. This space will be used to store data regarding the organization and management of the information contained in the disk (for example to keep track of which sectors contain data and which ones are still empty, the disk's directory name and the amount of free space on the disk, etc.). The 1541 DOS uses the track number 18 to store this information, thus reducing the available space for user data to 169984 bytes, corresponding to 664 tracks (683-19) multiplied by 256 bytes.
So, let us have a deeper look at the information stored in the track 18!

BAM - Block Availability Map

The BAM or Block Availability Map is the place where the 1541 DOS keeps track of the used sectors (the ones that already contain data) and of the free sectors (the ones still available to store new data). The BAM is stored in the

first sector of the track 18. For convenience, I have included here below the hexdump of the first part of the BAM (firsts 144 bytes) of the abacus cobol.d64 disk (please be aware that this is exactly the hexdump of my disk image, your disk image could be different).

```
00: 12 01 41 00 - 91396
01: 15 FF FF 1F - 91400
02: 15 FF FF 1F - 91404
03: 15 FF FF 1F - 91408
04: 15 FF FF 1F - 91412
05: 15 FF FF 1F - 91416
06: 15 FF FF 1F - 91420
07: 15 FF FF 1F - 91424
08: 03 00 02 0A - 91428
09: 00 00 00 00 - 91432
10: 00 00 00 00 - 91436
11: 00 00 00 00 - 91440
12: 00 00 00 00 - 91444
13: 00 00 00 00 - 91448
14: 00 00 00 00 - 91452
15: 00 00 00 00 - 91456
16: 00 00 00 00 - 91460
17: 00 00 00 00 - 91464
18: 10 EC FF 07 - 91468
19: 00 00 00 00 - 91472
20: 00 00 00 00 - 91476
21: 00 00 00 00 - 91480
22: 00 00 00 00 - 91484
23: 00 00 00 00 - 91488
24: 00 00 00 00 - 91492
25: 00 00 00 00 - 91496
26: 00 00 00 00 - 91500
27: 11 7F FF 03 - 91504
28: 12 FF FF 03 - 91508
29: 12 FF FF 03 - 91512
30: 12 FF FF 03 - 91516
31: 11 FF FF 01 - 91520
32: 11 FF FF 01 - 91524
33: 11 FF FF 01 - 91528
34: 11 FF FF 01 - 91532
35: 11 FF FF 01 - 91536
```

Still for convenience, to increase its readability, I have divided the above hexdump in groups of 4 bytes. The first number is the number of the track, whereas the number at the right is the absolute position in the disk of the fourth byte.

The first 4 bytes at position 00: represent respectively: the pointers to the track and sector of the first directory in the disk (hex 12/01, dec 18/01), the ASCII char A (hex 41, dec 65) which identify the disk format (in this case 1541) and an unused byte (00). What does this information mean? It simply means that the directory chain of this disk starts exactly at the track 18, sector 1. Let us keep this information away for now. We will come back here later.

The next 35 groups of 4 bytes store a schematic representation of the free/busy space on the disk for every single track. Now the level of complexity increases a little bit but I will try to make it simple.
In this specific example, I have decided to use the information contained in the track 18, because even if the disk is empty (but formatted) it will contain data.

```
18: 10 EC FF 07
```

The first byte (hex 10, dec 16) indicates the amount of free sectors in the track. The next 3 bytes indicate the map of space allocation for every sector of the track (in this specific case 19 sectors, see table 1). Storing the information using 1 byte for each sector would have been inefficient, so it was chosen to store this information in binary format: 0 means the sector is free, 1 means the sector is occupied. This explicitly means that we need to transform the data from byte (hexadecimal) to bit chunks (binary).

Firstly, we need to convert the values (EC FF 07) in binary format:
EC = 11101100
FF = 11111111
07 = 0111
If we put them in a single line, we will get 11101100 11111111 0111

Secondly, knowing that the bits are stored from the less significant to the most significant ones (little endian), we need to turn them all over (byte after byte…).
00110111 11111111 1110

The 3 bytes will look like:

```
16 - 00110111111111111110
     12345678901234567890 - position
                1        2
```

Finally, keeping in mind that 0 means the sector is free and 1 that the sector is occupied, we can count the occurrences of “1” values in the string. The result is 16, exactly as indicated by the first byte. Additionally we immediately notice that only the first, the second and the

fifth sectors contain data, the others sectors are free (empty). Wait a moment, the track 18 contains only 19 sectors (see table 1), what are we supposed to do with the "0" on the 20th position? We can simply ignore it because the 1541 DOS will not take care of it.

Here below all the 35 tracks of my abacus cobol.d64 disk. O=Free sector, #=allocated sector:

```
TK | FS | Sector space allocation
01 | 21 | OOOOOOOO OOOOOOOO OOOOO
02 | 21 | OOOOOOOO OOOOOOOO OOOOO
03 | 21 | OOOOOOOO OOOOOOOO OOOOO
04 | 21 | OOOOOOOO OOOOOOOO OOOOO
05 | 21 | OOOOOOOO OOOOOOOO OOOOO
06 | 21 | OOOOOOOO OOOOOOOO OOOOO
07 | 21 | OOOOOOOO OOOOOOOO OOOOO
08 | 03 | ########## #O######## #O#O#
09 | 00 | ########## ########## ######
10 | 00 | ########## ########## ######
11 | 00 | ########## ########## ######
12 | 00 | ########## ########## ######
13 | 00 | ########## ########## ######
14 | 00 | ########## ########## ######
15 | 00 | ########## ########## ######
16 | 00 | ########## ########## ######
17 | 00 | ########## ########## ######
18 | 16 | ##OO#OOO OOOOOOOO OOO
19 | 00 | ########## ########## ####
20 | 00 | ########## ########## ####
21 | 00 | ########## ########## ####
22 | 00 | ########## ########## ####
23 | 00 | ########## ########## ####
24 | 00 | ########## ########## ####
25 | 00 | ########## ########## ##
26 | 00 | ########## ########## ##
27 | 17 | OOOOOOO# OOOOOOOO OO
28 | 18 | OOOOOOOO OOOOOOOO OO
29 | 18 | OOOOOOOO OOOOOOOO OO
30 | 18 | OOOOOOOO OOOOOOOO OO
31 | 17 | OOOOOOOO OOOOOOOO O
32 | 17 | OOOOOOOO OOOOOOOO O
33 | 17 | OOOOOOOO OOOOOOOO O
34 | 17 | OOOOOOOO OOOOOOOO O
35 | 17 | OOOOOOOO OOOOOOOO O
```

In the *Table 2* there is the meaning of each byte of the first part of the BAM.

Is the BAM completed? Almost, but at its end there are other 27 significant bytes which contain useful information. Let's have a close look at them in *Table 3*.

Position	Byte	Meaning
0-1	2	Position (TS) of the first directory
2	1	Disk format (hex 45 = 1541)
3	1	Unused
4-7	4	Map of the first track
8-11	4	Map of the second track
...	...	Map of the next tracks...
140-143	4	Map of the 35th track

Table 2: First part of the BAM

Position	Byte	Meaning
144-159	16	Disk name, filled with spaces
160-161	2	Spaces
162-163	2	Disk ID
164	1	Space
165-166	2	DOS version and format (2A)
167-170	4	Spaces

Table 3: Second part of the BAM - Disk name, Disk ID, version

All this information can be read and displayed converting the Hex value in the corresponding PETSCII char (for convenience, in the example attached below I used the ASCII instead. It is almost fully compatible with the only exception of some special characters).

The Directory chain

Now that we have finished reading the BAM, we need to print out the list of the files contained in the disk. Where is this information stored? It's still on the track 18, but since the Disk Drive 1541's engine runs at 300 rpm in order to optimize the performance the 1541 DOS distributes the directory chain in a non-contiguous way among the sectors.

In general the directories in a disk are distributed following the sectors' sequence below:

0 (BAM), 1, 4, 7, 10, 13, 16, 2, 5, 8, 11, 14, 17, 3, 6, 9, 12, 15, 18.

Bear in mind that this is just the standard distribution. During my tests, I have found out that most of the disks follow this structure, but I also found some exceptions. How can we avoid errors given by the exceptions? Easy, we just need to follow the indications provided by the disk

itself and disregard the above distribution!

The only certain information we know is that the pointer to the first directory in the disk is located in the first 2 bytes of the track 18; if you remember, we kept this information away in the first part of the article: 12 01 41 00.

Well, that information tells us that the first directory is located in the sector 01 (the second byte) of the track 18. From there onwards every directory will contain in its first 2 bytes the pointer to the next one. Therefore we just need to follow the bread crumbs and we won't get lost... hopefully! So let's follow them until we won't find in those 2 bytes the values (hex 00/FF, dec 00/255) indicating that we reached out the last allocated directory in the disk (see *Table 4*).

Position	Byte	Meaning
1-2	2	Track and sector of the next directory: 00/FF indicate we reached out the last allocated one.
2-31	30	First file in the directory
32-33	2	Unused
34-63	30	Second file in the directory
64-65	2	Unused
66-95	30	Third file in the directory
96-97	2	Unused
98-127	30	Fourth file in the directory
128-129	2	Unused
130-159	30	Fifth file in the directory
160-161	2	Unused
162-191	30	Sixth file in the directory
192-193	2	Unused
194-223	30	Seventh file in the directory
224-225	2	Unused
226-255	30	Eighth file in the directory

Table 4: Directory structure

Now that we know how to follow the directories, we just need to read the information contained which will represent the file name, the file format, the file size and the position where the file data are stored in the disk (at least the beginning of the file).

This information is contained in 30 bytes with the following structure (see *Table* 5).

Nothing really complex, except for the DOS File type that needs to be converted using information in the *Table 6* (see below) and the file size which needs to be calculated keeping in mind the LO/HI byte method (little endian).

Position	Byte	Meaning
1	1	DOS File type (see table 6)
2-3	2	Track and sector of the beginning of the file
4-19	16	File name (beware of PETSCII chars)
20-22	3	Unused except for relative files
23-26	4	Always unused, set to '00'
27-28	2	Reserved by DOS for file operation
29-30	2	File size stored as little endian

Table 5: file directory description

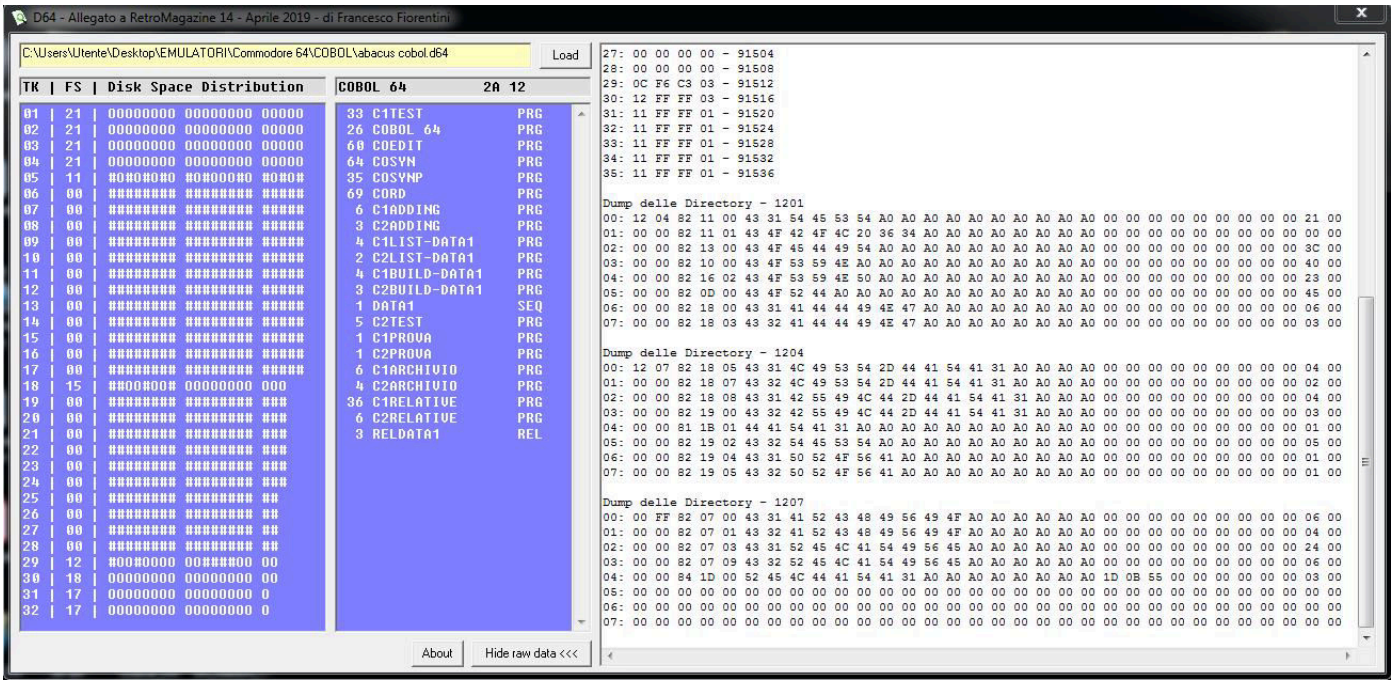
HEX	DEC	Definzione
$00	0	Scratched - does not appear
$80	128	DEL - Deleted
$81	129	SEQ - Sequential
$82	130	PRG - Program
$83	131	USR - User
$84	132	REL - File Relative
$00	0	Scratched - does not appear
$01	1	SEQ - Sequential (unclosed)
$02	2	PRG - Program (unclosed)
$03	3	USR - User (unclosed)
$04	4	REL - Relative (unclosed)
$AO	160	DEL - Deleted (replacement)
$A1	161	SEQ - Sequential (replacement)
$A2	162	PRG - Program (replacement)
$A3	163	USR - User (replacement)
$A4	164	REL - Relative (replacement) - uncommon
$CO	192	DEL - Deleted (locked)
$C1	193	SEQ - Sequential (locked)
$C2	194	PRG - Program (locked)
$C3	195	USR - User (locked)
$C4	196	REL - Relative (locked)

Table 6: DOS file type

What does 'LO/HI bytes' (little endian) really mean? It means that, in order to calculate the file size, we need to sum the leftmost byte (low-byte or the less significant) + the rightmost byte (high-byte or the most significant) * 256. In this way the 1541 DOS can save precious bytes to store information (don't forget, they were designed in the 80's and every byte counts).

The resulting formula therefore is: file size = lo-byte + hi-byte * 256

Let us have an example with the data as in Figure 2. The values are 21 00, then using the above formula:



2. The Visual Basic 5 program coded to test the .D64 file format

File size = 33 (21) + 0 (00) * 256

In total: 33 used blocks on disk by the file named “C1 TEST”. As you may have easily understood, the track 18 of the D64 and consequently of the 1541 diskette format is the starting point to access the information contained in the C64 disks. Why this specific track and not for example the first one or the last one?

I bet because the track 18 is located exactly in the middle of the disk and starting from there the DOS can quickly access every other track optimizing the read/write performance with just a short movement of the header. Smart decision by the designers, indeed.

I hope I have been able to provide enough information on the D64 file format and on the basic functionality of a real 1541 disk. With this little knowledge, everyone should be able to create its own utility to read the basic data from a D64 file. In case of questions, you can contact me or anyone else in the RM team.

During the writing of this article, I took the opportunity to code a simple Visual Basic 5.0 program just to validate the correctness of the information provided.

The source code and the binaries can be download from: http://www.retromagazine.net/getrm.php?id=d64

The logic of the program is quite easy. It opens a D64 file in binary mode and reads every single byte by starting from the first byte of the track 18 (the 91393rd byte of the file, obtained multiplying the amount of sectors in the first area, 657, by 256 and adding 1). Then using the information provided by the tables shown in this article, the program decodes the data and display them in a text field. Hope you like it, it works quite well and it is reasonably fast, but of course it can be improved! Maybe you want to give it a try.

You can also download Visual Basic 5.0, released by Microsoft in 1991, from Winworldpc.com: https://winworldpc.com/product/microsoft-visual-bas/50

Interview with Gideon Zweijtzer

The man behind the U1541 II+ and the Ultimate 64

by David La Monaca (Cercamon)

Hi, Gideon and thank you so much for accepting my invitation for an interview. All the readers and the editorial staff of RetroMagazine are very excited to have the opportunity to ask you some questions about your experience in designing one of the most (if not the most) famous cartridge/expansion for the C64: the 1541 Ultimate. During 2017, according to your web site, the final steps of the Ultimate64 design have been completed, so the long-awaited board has finally got into production and the first batches have been shipped to the final users earlier this year.
Most of the Commodore 64 fans out there are well-aware of your fantastic products, but I'm pretty sure they don't know how it all began. So let's start from the beginning.

Can you please shortly introduce yourself and tell us something about your own story (ie. where you are born, growing up, your education, your personal interests, etc.)?

Hi David, thanks for the invitation! Talking about myself? Sure... I was born in Amsterdam in 1974, in a quite stable family with one older brother. I have always been interested in technicalities. Before the home computers came, I was always with my technical Lego, although I also loved to race around on my bike through the neighborhood. I often played with circuits made from switches, motors and light bulbs, but unfortunately I did not have anyone in my surroundings with knowledge of electronics. From the secondary school, I went to the university TU Delft, where I studied Electronic Engineering.

I guess you have always been a computer fan and user since when you were a kid. What started you on the path of computing and what was your first experience with a computer? Was the Commodore 64 your fist computer?

I was pretty young when we got an Atari 2600 game console. It was actually my brother who had started with the whole computer thing and he started to investigate the possibilities for programming. There was a basic interpreter for the A2600 at that time, but in the end he bought a ZX81. I was not really allowed to touch it, but sometimes I sneaked into his room and tried a few things, but as a kid without any help, I didn't get that far. Later, my brother got a Commodore 64 and this got big. It was so popular in that time! Computer clubs, meetings, copying games and programs! My brother infected me with his curiosity about programming and although he didn't want me to bother him, I could sit on the floor in between his massive desk and the old color TV that was on top of another table in front of that. As long as he didn't hear me, I could just watch what he was doing. I saw basic, assembler, etc. At a certain point I could tell him from behind the desk that he forgot a statement... It was not until I reached the age of 11 that I got my own Commodore 64.

Gideon in his living room

How did you get started working on a C64, beyond playing games? Did you quickly find interest in programming and discovering how the machine intimately work?

I never really played a lot of games, actually. There are some exceptions, like Giana Sisters. But in general, I did not spend a lot of time on games altogether. As my brother focused a lot on the software, my interest in the hardware grew. At a certain point I made a simple thermometer, using an NTC thermistor on the paddle port of the C64. I needed my older brother again for his math skills to figure out the conversion curve. At the computer club in Amsterdam, I usually spent my time around the repair stand, where I could see how some guys were de-soldering and replacing chips in broken C64's. According to my mother, I had the full schematic of the C64 hanging from the wall in my small bedroom. But in all honesty, I don't remember that.

After the C64, did you get your first PC and still keep the C64 on your desk? Did you ever use one of the many SD2IEC devices on the market before starting to design the 1541 Ultimate?

No, I haven't. In fact, I think you're now skipping quite a few years. My interest in the C64 faded as the Amiga 500

came, and later the PC. Actually my first PC was a Pentium 120 MHz, so you can imagine that I have resisted PCs for quite some time... The love for the C64 never really went away, I just never used it. Neither have I ever been part of a demo- or game coder group, or the "scene" in general... So there was basically never a need for an SD2IEC or any other C64 peripheral.

What inspired you to design the first version of the 1541 Ultimate and when did you start?

The first version of the 1541 Ultimate was made in 2007. It all started with some implementations of the 6502 as I was learning and getting more experience with FPGA design using VHDL. That was back in 2001 or so. There were a lot of things going on back then. For instance, Jeri Ellsworth was working on her C-One, which later became the DTV, if I am not mistaken. In any case, I had already done a lot of the C64 in FPGA at that time, but I didn't see the point of being a "me-too" player. So I thought I'd do the floppy drive instead. On one of the club meetings that we have in Maarssen, I demoed the very first prototype on a Xilinx Spartan 3 board. You needed a laptop or PC to download a floppy image over Ethernet into its memory, after which the board acted as a floppy drive. No menu, no other emulations, only the drive. Later, in a conversation with one of my colleagues at work, the idea arose to build it into a cartridge, such that the VIC could be used to display a user-interface. This idea crystalized in 2007.

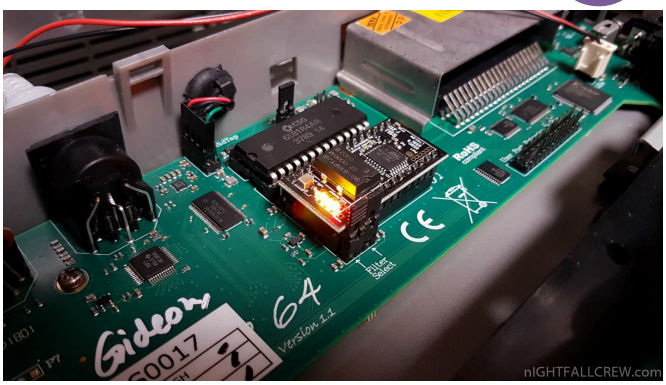
Did you design the hardware and the software/firmware for the 1541 Ultimate all by yourself?

Yes, basically. There have been some important contributions from others over the years, though. But in essence, the hardware design, the FPGA design and the firmware design and framework are made by me.

What was your computer system setup that you used to develop and test the early project of the cartridge?

Just a PC and one Commodore 64... And yes, that did not include a 1541 drive! Later it showed that this was not enough, but I did not have more hardware, so I visited some friends from the Commodore club that had impressive collections of machines to test the compatibility with. In fact, there I found out that the very first prototype of the 1541 Ultimate as a cartridge was not very compatible, which caused some design changes before the board went into production.

The cartridge 1541 Ultimate II+



Close view of the Ultimate 64 v1.1

Did you take any computer courses to start you in the field of electronics? And if so, what were they and how much time did you invest? Or, like many designers/ programmers of the early Eighties, were you a self-taught techie?

Many things were self-taught, although studying at Delft University of Technology has made me understand many more things. But to be fair, I think that I learned most at the job after my studies. I started to work as a junior designer at Technolution B.V., and there I learned most of the practical knowledge that I have today, in terms of electronics design. Interestingly, I brought knowledge about FPGA design back as I was one of the founders of this discipline within the company.

What was your development process like? Did you use to sketch out concepts, design the mainboard and the firmware, etc.? Do you still take on the design process the same way?

Oww, that's quite a difficult question. Because I have always seen these activities as a hobby, I mostly just let it happen. I am the kind of designer that does a lot of design work 'as a background process'. I am not a very structured, method-following, step-by-step kind of engineer. (I made quite a few project-managers pull their hairs out, as they didn't see me work on new tasks they assigned to me in the first weeks...) I work with iterations, basically. But mostly just in my mind. Sometimes under the shower, or while driving. Once it 'feels right', I start to do some implementation. And sometimes after an implementation I realize it doesn't feel as right anymore. I am not afraid of just throwing some work away and start anew. Of course, always taking into account the lessons learned in the previous step.

Talking about your biggest projects (the 1541 Ultimate cartridge and the new Ultimate64 mainboard), what technical challenge gave you the biggest feeling of accomplishment?

Ok, when I need to limit it to 'technical challenges', it would definitely be the solving of very hard-to-find bugs... You know, those nasty ones that make others quit on their project... Those! On a second place, it is when I power up a new board and everything works right away. (And that's not uncommon in my case... [smug face]).

Gideon's logo on the Ultimate 64

What was the biggest tech/programming obstacle that you ever overcome while designing/producing/testing/selling the 1541 Ultimate or the Ultimate64?

Obstacles... [thinking]... It depends a bit on how you define the obstacles. Most things are just time consuming tasks. But yet, I think there are several 'obstacles'. I think in case of the 1541 Ultimate, it must have been creating an easy to use user interface without having access to any framework; building everything from scratch. On an embedded platform, which the Ultimate clearly is, you can't use standard frameworks like the ones commonly used in Java and C#, so you have to make one of your own. Hmm, another obstacle was the development of a factory test system for the Ultimate-II+. That took quite some time. But then, I do think it saves me a lot of time. Another one was the move to a web-shop system, rather than just taking orders and processing them manually.

What was/is your favorite game for the C64? Do you still find some time to play?

Giana Sisters... err.. no time to play!

I imagine that you do own a collection of stock C64s (i.e. all versions: from C64 "breadbin" with all the ASSY board revisions, to C64c, C64g and C128) for testing purpose. Are you a collectionist of retrocomputers as well, not only Commodore branded?

My wife would kill me, if I were actually collecting more. I only have working C64 mainboards, of which I use mainly just one in a C64C case. This has been the same machine as I used to test over 3000 ultimate's over the years. The power switch and cartridge port are a bit sad now. I do have a C128 and a C128D, but I never use them. I do have several floppy drives, too.

Can you even think about calculating how many hours you spent designing and working on the several versions of the 1541 Ultimate cartridge? What about the Ultimate64?

It is very difficult. As I said, many design activities take place as a background task. If I would count only the hours that I spend on the PC it might give a falsely low figure. What I can tell, tho, is that hardware designs, board layouts and such, usually don't take that much time. I think I created the U64 board design in about 3 weeks' time, but then of course only in the evenings and weekends. The schematics took a similar amount of time. Most time spent on technicalities goes into FPGA design, implementation and debug and firmware implementation. From your question I sense that you focus a lot on the technical aspects, but I can tell you that the administrative tasks, including shipping orders and answering e-mails takes up most of my time, unfortunately.

Have you ever worked or are you planning to work on other projects involving the C64 or even different 8/16-bit machines?

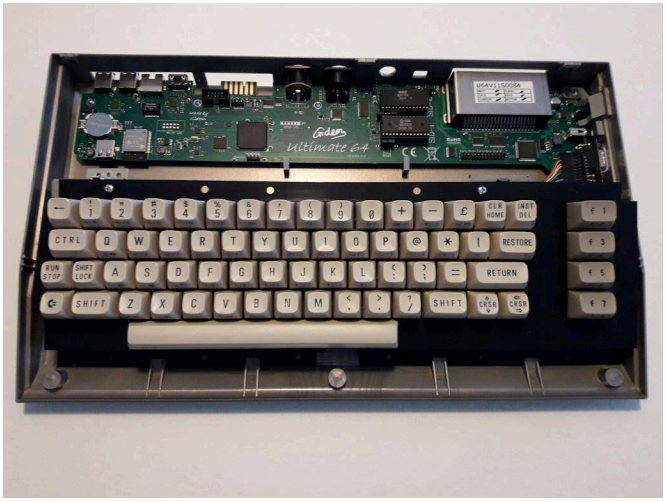
Nope... :-)

How many people currently work at Gideon Lab on producing, testing and selling the two main products? Did you ever work in a team or simply get consulted with other electronics/software experts in order to achieve a particular result or to solve a bug?

Production is outsourced to a number of companies. (Production-) testing of the Ultimate-II+ is also performed in the factory. Production test for the U64 doesn't exist yet as of today, but that will be the next step in order to accelerate the process. When we talk about assembling the U2+ into plastic cases, that's often done by my wife, ... when she feels like it. She also plays an important role in packing orders. The other things are done by me; there are no employees at this point. Whether this can continue like this, is questionable. I think I do need external help for the quantity of U64's that are currently on order. On the technical aspect, I sometimes talk with my colleagues about certain bugs, and of course I use the feedback and input from the community. There are some pretty smart guys out there that help me solve bugs sometimes. In order to achieve a particular result, I often apply patterns that I quietly pick up or learn from other projects.

Looking back to where you started it all, is there something that you regret about the PCB design or any other detail? Would you do something in a different way now if you could?

I mostly regret not taking the C64 FPGA code that I had made years before the U64 to a production level. I actually



The Ultimate 64 installed in a Commodre 64 case

The Ultimate 64 motherboard in all its glory!

demoed a complete C64 in FPGA already back in 2011. I thought nobody would be interested in buying an FPGA-based C64 motherboard, since original C64 machines could be picked up for almost nothing, or else people would use an emulator anyway. Regrets about other aspects: well, in retrospect many things could be regretted. But I think it is not fair to look at things like that, because as a person and as an engineer, you learn while you do it. Once you think things have to change, there is always the freedom to do so. I think that is one of the very cool things about having your own product. But I guess this principle applies to many things in life, doesn't it..?

I'm pretty sure that you worked very hard on both your projects during the last few years but also that you had so much fun doing it. What is the most funny/weird moment or story that you've been through while developing your products?

Oh, I absolutely had much fun doing it! Technically speaking, I have the most fun doing the FPGA code, second the hardware itself, and third the firmware. I think one funny moment was the moment I realized how naive I can be. In the whole process of creating the 1541 Ultimate, I *never* thought of actually making a sellable product out of it. Or let's say, that was not my goal; it had always been pure hobby until then. It was actually a Swedish scener, TwoFlower, who happened to visit the Commodore Club in Maarssen just when I was giving the demo of a cartridge with an embedded floppy drive. He said I should have it produced, but I was hesitant and thought that it was not even feasible to do so. He asked me how many needed to be produced, and I stammered, "maybe 40 or 50?" He smiled and said: "Just do it... I'll make sure you'll sell all 40 of them in Sweden alone!" And that's how it all started!

Gideon, thank you very much for your time. This interesting interview ends here. Would you like to add anything, or say anything to our readers?

There is one important thing to mention.. I would like express a huge 'thank you' to the Commodore loving community. One of the most rewarding aspects of this project is the great feedback, the positive words I receive. In short: without you guys, I would never have been able to do all this. Thank you.

External links

• **1541 Ultimate** Cartridge's official site:
http://www.1541ultimate.net/

• **Ultimate64** Motherboard's official site:
http://www.ultimate64.com/

One of the Commodore Club's meetings in Marseen (The Netherlands)

Sinclair QL: mistakes, misfortune and so many regrets

by Alberto Apostolo

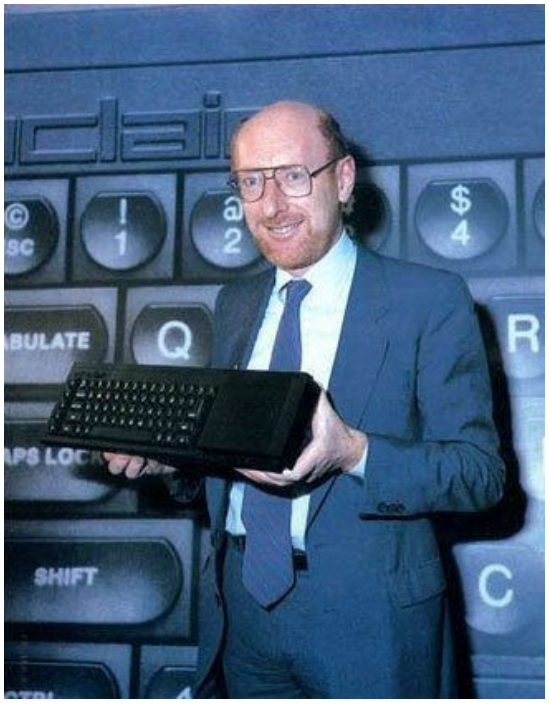
This article completes the tetralogy on the machines produced by Clive Sinclair, started with MK14 (RM 6), continued with the calculators (RM 9) and the series of ZX computers (RM 11) (At the moment I'm writing they are available in Italian language only, sorry).

In 1982 Clive Sinclair started designing the QL: a computer for management use more advanced than the newly launched ZX Spectrum. The hardware consisted of a Motorola 68008 processor with 128 KB RAM and two 100 KB microdrive readers (not compatible with the ZX Spectrum). The SuperBasic and QDOS operating system working in multitasking were available on the machine.
In addition, a wealth of application programs made by PSION were offered: Abacus (spreadsheet), Archive (database), Quill (word processor), Easel (business graphic).
To counter the launch of the Apple MacIntosh, the QL was presented early on January 12, 1984 in London, proving to be a resounding failure that highlighted the difficulties of a company that grew too fast.

The Sinclair QL computer

The laborious development of the computer, the incomplete release with serious delay, the price of £399 were fatal for QL and Sinclair Research itself (also exposed in the production with very few sales of a flat-panel TV and the C5 electric tricycle).

In 1985 he closed Timex (the American manufacturer of Sinclair licenses) and went into crisis Prism (the mai distributor in the United Kingdom). Now on the verge of bankruptcy, Clive Sinclair was forced to sell his products. In 1986 Amstrad (Alan Michael Sugar Trading, n.o.s) purchased the computer branch from Sinclair Research for GBP 5 million.



Sir Clive Sinclair

This operation decreed the end of the production of Sinclair QL (estimated at about 150,000 copies in total). Amstrad continued the evolution of the ZX Spectrum until 1990, producing the +2 and +3 series, despite the presence of systems with superior performance such as MSX, Commodore Amiga and Atari 520 ST.

Clive Sinclair returned to computer science in 1988 with the new Cambridge Computer brand and ZX-88 battery laptop equipped with LCD display and Z80 CPU. While it made good impressions, it quickly disappeared because the market had definitely changed due to compatible PC-IBM and the demand for higher quality hardware and software.

The Cambridge Z88 portable computer

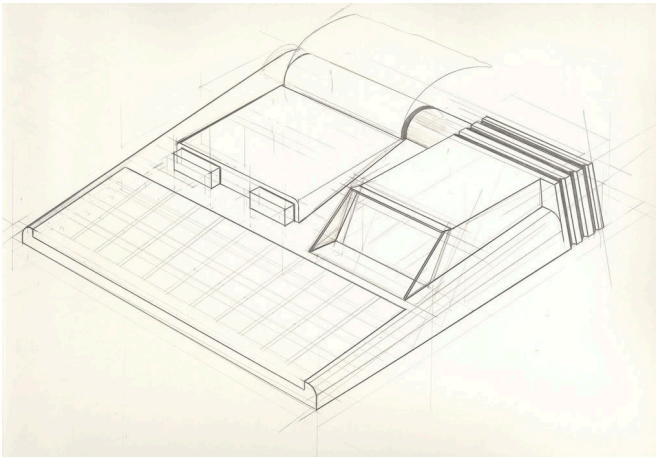
A Late summer dream of a professional computer

In 1981, 36-year-old Robb Wilmot was hired as CEO by I.C.L (International Computers Limited, a recently privatized British public company). He had previously been the youngest vice president of Texas Instruments (he headed the Calculators Division). He was also a friend of Clive Sinclair and believed that Sinclair technology could revitalize the range of office computers.
In addition, Dataskill (a company controlled by I.C.L.) already offered programs for the Sinclair ZX81 and had produced management software for home microcomputers since the time of Nascom 1.

Against this background, an agreement between Sinclair and Wilmot was announced in December 1981 as follows: (1) I.C.L. would be licensed by Basic Sinclair (a move made in response to British public television B.B.C. which had adopted the technology produced by competitor Acorn), (2) I.C.L. would buy the flat-screen TV produced by Sinclair to be combined with a possible office computer that could also be used as a communications terminal (Sinclair would develop the hardware of such a machine), (3) Sinclair would obtain one million pounds to finance development costs and would also obtain royalties if the machines were distributed to customers.

Between May and August 1981, Rick Dickinson (former designer of ZX computers) had produced some sketches depicting a keyboard machine, a pair of Microdrives (the tape system developed by Sinclair) and a small built-in display. Other sketches also showed a built-in printer. They were not far from what the One-Per-Desk, the system desired by I.C.L., should look like.

In May 1981 the Osborne 1 laptop appeared on the market; however, Dickinson's sketches do not give the idea whether the computer he imagined was a laptop or not.



One of Dickinson's sketches

Shortly after the launch of the ZX Spectrum, Clive Sinclair stated in April-May 1982 that the next step would be a new computer in an appropriate price range higher than the Spectrum, based on the same philosophy as Osborne 1, more manageable than IBM system, weighing 1-1.5 kg, with a printer and flat-screen TV connection.
David Karlin, hired by Sinclair in the late summer of 1982 as Chief Project Engineer of the computer division, is in charge of developing a project. David Karlin (at the time just over twenty) had obtained a bachelor's degree and specialization in Electrical Engineering in Cambridge. He then spent some time working at the Xerox in Palo Alto and the Fairchild Camera and Instrument Corporation in Singapore.

Rick Dickinson in his design studio

During the Xerox period, he had been able to observe the Xerox Star and its W.I.M.P. system (Windows, Icons, Menus, Pointer) which had influenced personalities such as Bill Gates and Steve Jobs in making their products. When he later decided to return to the UK for family reasons and seek another job, he was not thrilled to work at Cambridge. However, during an interview with Sinclair, Karlin had exposed his vision of a £500 Xerox Star system and Clive Sinclair had convinced him to take up a job (offering him the same salary as Fairchild, which was above the UK average at the time).

Great expectations for the ZX83

David Karlin immediately set to work to write the specifications of a management machine called ZX83. It was meant to be a desktop computer with what he thought was the bare minimum: a decent keyboard, some kind of network connection, a dedicated monitor, and a printer. Interviewed, he said, “In fact, in terms of broad spec, you could say I was trying to design what the Amstrad PCW

8256 would eventually become."

It also had to have some sort of window user interface, which would require: high-resolution bitmap graphics, enough memory and a fast processor. Intriguingly, he did not consider the idea of equipping the system with a mouse. A choice that seems strange today but at the time he was fairly confident that people could do enough with arrow keys.
Efforts then focused on circuitry with the priority of saving as much as possible.
Since the Z80A microprocessor addresses only 64 KB and memory paging techniques would have limited its performance, including the need for a 16-bit processor. After the Zilog Z8000 and Intel 8086 were discarded, the choice fell on the Motorola 68000 processor series considered very good and a great platform for the future. It was decided to adopt 68008 at 7.5 MHz, about twice the frequency of Z80A used on the ZX Spectrum. Internally, it worked 32-bit with an 8-bit data bus and 20 bits reserved for addressing.

The 68000 (which will be used on the first Apple MacIntosh, the Amiga and the Atari 520 ST) also worked internally at 32 bits but with a 16-bit data bus, 24 bits for addressing and a cost, at the time, 2-3 times higher than the 68008. Based on Karlin's recommendations, Clive Sinclair in December 1982 approved the use of the 68008, practically gambling the future of the company on that platform, believing that at that time even the competition could not afford to produce a computer based on the 68000. Unfortunately, in 1983, Motorola cut the price by 68,000 below the price Sinclair had contracted for 68008.

Renegotiating the contract and adopting the 68000 would not have been onerous, but adding the 68000 in the architecture would have required extra benches of ROM and RAM and a separate chip for the system logic (given the evolution of the project, it would have been better this way but it is too easy to talk in hindsight).

The architecture of ZX83 included some chips for ROM, 64KB RAM (32KB for programs, 32KB for video memory) and two U.L.A. chips (Uncommitted Logic Array) called ZX8301 and ZX8302.

The ZX8301 would connect the 68008 to the rest of the system; it would function as a processor clock, memory timer, and mediator between the 68008 and the display controller to access RAM. The display would be checked via a dedicated connection.

Jan Jones

The ZX8302 would manage the Input/Output including Microdrives, Keyboard , Network, Printer Port. It would incorporate a modem (as requested by I.C.L. for the One-Per-Desk above) and A R.T.C. (Real Time Clock) powered by a battery.

In early 1983, after choosing CPU and defining a basic scheme of the system, Karlin began to evaluate the machine based on these components and to specify the basic software required.
At the same time, Tony Tebby and newly hired Jan Jones were commissioned to develop the basic software: the operating system and BASIC Interpreter.

Prior to joining Sinclair in 1982, Tony Tebby was a Physics graduate who had learned programming on his own while working at G.E.C. on microwave systems. He had done so because he thought the software used by that company was of little value and had managed to create something better. This changed his career, first at Philips and then (in 1979) at the Computer-Aided Design Centre in Cambridge, a joint venture between I.C.L. Dataskill and the British Ministry of Industry and Commerce. It was at C.A.D. that Tebby met Jan Jones, a programmer who had previously graduated in mathematics while working for other companies.

It was Tebby himself who convinced Jones to join Sinclair and collaborate on the project to create the SuperBasic, a version of the Basic Sinclair enriched with structured programming elements that were already included in the Acorn BBC's Basic.
Jan Jones had to perform the encoding using the 68008 Assembler after a lengthy analysis process to define the commands with their characteristics.
The intentions were to build the language "inside" the machine, not only to program or allow third parties to create applications, but also as a shell language for "elegant, easy to use and full of functions" operating commands. With this, the Sinclair leadership wanted the "core" of ZX83 to become the basis for the future "incarnations" of ZX Spectrum.

David Karlin intended to use Tebby's QDOS called "Domesdos". If it didn't work, it would have folded back into an operating system built by G.S.T. Computer Systems. According to Tebby, on the contrary, G.S.T. was contacted (without the approval of the Sinclair management) but could not finish it in time. However, there are several indications that Karlin's version is the real one.
Attempts to involve Digital Research and Microsoft were also noteworthy, but their achievements were not good for the hardware of the time.

In December 1982, Nigel Searle had contacted potential partners to develop the software to be sold with the computer. Among the companies contacted was PSION (which will later produce Symbian, a mobile operating system). David Potter, was the ambitious director of PSION who had understood the importance of management software to grow the company and compete with Americans in the word-processor and spreadsheet market.

After long discussions, PSION was chosen without involving Karlin and Tebby (however, packaging the applications with the computer was part of the plans). PSION claimed that the QL could support an 80x25 character screen and established Quill as a word-processor, Easel as a graphical tool, Archive as a database and Abacus as a spreadsheet. These programs would in future constitute the "xChange" suite, a kind of de facto standard for 16-bit computers (developing an integrated suite entailed fewer risks than developing several "stand-alone" applications).

Each application was entrusted to a team-leader: Martin Brown for Easel, Martin Stamp for Quill, Charles Davies for Archive, Colly Myers for Abacus (the latter became General Manager of PSION and later CEO of Symbian). PSION programmers will take 15 months to complete applications. Without any idea how ZX83 worked, they used a VAX micro-computer with a 68008 emulator until ZX83 with QDOS was available.

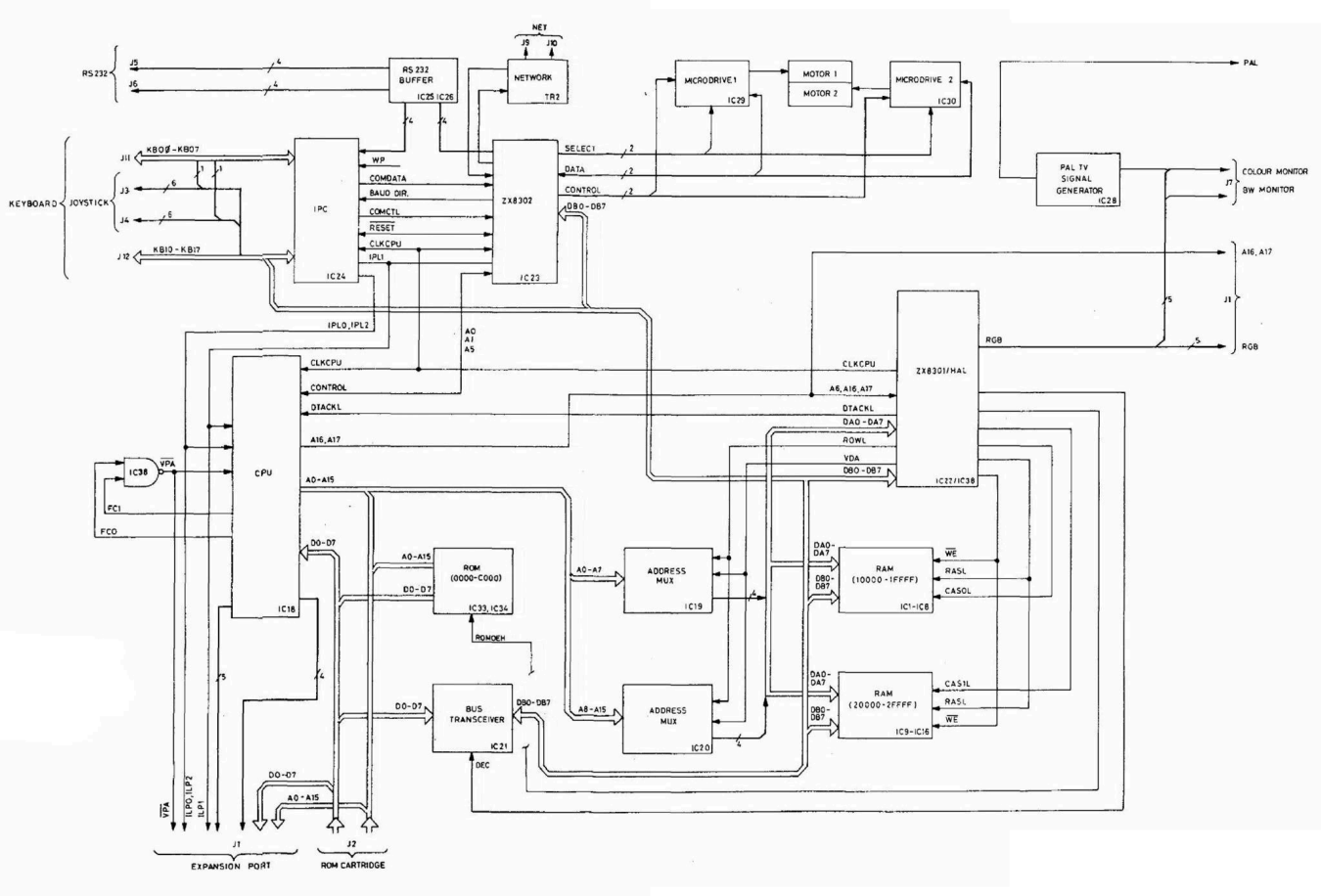
The dream soon becomes a nightmare

In March nine months were established as the duration of the project, to be sure to launch the product shortly before Christmas.
A crazy deadline: Sinclair had never been able to create a machine in such a short time and, among executives, it seems that only Jim Westwood (Chief Hardware Engineer) had highlighted the need to take more time for development (without however being listened to). Also, there was no clarity in management about what ZX83 really was: was it a laptop or a desktop computer?

Nigel Searle (CEO of Sinclair)

In May, Nigel Searle (CEO of Sinclair) called a press conference to publicly announce the development of the new ZX83 model, stating that it would not be a clone of PC-IBM and that Sinclair would develop its own operating system.

Moving away from operating systems such as the Digital Research CP/M or Microsoft's emerging MS-DOS (which were in fact standards at the time) was a real gamble. In front of an enthusiastic audience hearing that "Uncle" Clive Sinclair was about to place another hit after the ZX Spectrum, Searle teased participants with the news that



The electric scheme of the Sinclair QL

the ZX83 could be a laptop with a display based on Sinclair flat screen TV technology and solid-state memories.
He also leaked that the car would probably be called QL, an abbreviation for "Quantum Leap" and that the launch would be scheduled for January 12, 1984 at the Inter-Continental Hotel at Hyde Park Corner in London.

The last hopes of having a laptop lasted until the Summer of 1983, following the discovery that the batteries produced for the Sinclair Microvision 2700 pocket TV would guarantee only 30 minutes of autonomy to a laptop (10 minutes with the Microdrives connected together) and that a text displayed on a Sinclair micro-display (connected for trial to a ZX Spectrum) was barely readable.

Putting so many features on only two 40-pin chips had proved almost impossible (serial I/O required 8 pins and not 4) and the idea of creating a new ZX Spectrum based on ZX83 was quietly put aside between August and early September 1983.

The ZX8302 was not complete and needed to be lightened by the keyboard interface. This led to the most controversial decision within the project: to include an Intel 8049 micro-processor in the system logic. The Intel 8049 was improperly dubbed the Intelligent Perpherals Controller. It was also able to generate sounds and was useful for lightening ZX8302 of serial port management.

This resulted in negative side effects. The 8049 had its own state registers and Karlin could no longer use a centralized bank of state registers and peripheral interrupts (a "clever" scheme he devised that allowed him to save on hardware costs and simplify software).

So QDOS could also handle 8049, Tebby called Aaron Turner. Although it was hard work, Turner succeeded.
Instead, it was much more difficult to place the 8049 and its connecting lines on the long, narrow motherboard, derived from the shape of the houselong chosen for the ZX83. The problem of motherboard measurements got worse when there was the move to make ZX83 more "home", also adding a TV output and joystick connections (strange accessories for a management micro). The joysticks were added thanks to the 8049, while UHF modulator caused large headscratches because it had been placed on the right, near the Microdrives and their fragile read/write heads, sensitive to the electromagnetic disturbance of the oscillator. It was also decided that it would take the "classic" departure of the Basic at the time of ignition.
To make the SuperBasic, Jan Jones had worked hard and respected deliveries, but the SuperBasic was not designed

to turn "over" QDOS.

Aaron Turner easily adapted the graphics routines of the SuperBasic (created by G.S.T. in an attempt to buy time) to work through QDOS Display Manager instead of writing directly into RAM reserved for the screen. He rewrote the Interpreter so that he could get/release memory through QDOS Memory Manager instead of doing it himself.

Four weeks before Christmas, Tebby and Jones used most of the pre-Christmas launch session to test the software's interaction with hardware. Tebby decided to "dismantle" QDOS and SuperBasic to allow the Interpreter to shoot as a special privileged monolithic task. A serious error that compromises the integrity of the operating system.
At the same time, there was still the question of the double version of the SuperBasic: one compact and minimal wired in ROM and another larger one loaded through the cartridge of a Microdrive (the "minimal" Basic should have noticed the "extended" Basic and loaded the extensions accordingly). The "fairy tale" of the two Basic's would later have caused an unpleasant misunderstanding.
At the end of 1983 it became clear that the QL was behind schedule. The blame was due to doubts about what ZX83 was supposed to be and to Sinclair management refusing to extend the project's deadline. Deciding that ZX83 management platform should be the basis for future versions of the ZX Spectrum had not only forced a new version of the SuperBasic based on the 68008, but had forced Karlin to revise the initial specifications for the two ULA chips.

For example, the display controller circuitry in ZX8301 needed to be modified to support the existing modes on the ZX Spectrum.
The ZX8302 had required a suitable sound generator for the machine to produce better sounds and music than the ZX Spectrum.
Separately it was decided that it would take too long to implement a modem inside ZX8301 and the modem was deleted. Similarly, thededicated output port for the printer was deleted, replaced by two generic RS232 ports.

Misleading advertising and the "thriller" about the external ROM

The project deadline had been formally modified, moving it to the beginning of 1984.
David Karlin and Jan Jones would need another six months of work due to the lengths due to continuous revisions of the printed circuit boards and test ULAs. According to Tony Tebby, they still didn't have a fully functional prototype available.

David Karlin should have warned Nigel Searle and Clive Sinclair, but the January 1984 deadline for press releases had nevertheless been maintained. This state of affairs displeased Tebby. The straw that broke the camel's back was the announcement (during the presentation) that customers would have the machine within 28 days of ordering and that orders would be accepted from the end of January. Tebby announced that he would leave Sinclair when the QL was in production, which happened promptly in April.
The computer was offered with 128 KB RAM at the price of £399 plus £7.95 for shipping costs. No mention of the 64 KB "cheap" version of RAM at the price of £299, as PSION applications required more than 32 KB RAM each.
In the United States (where Searle founded a subsidiary of Sinclair in 1980, then operated according to Cambridge since 1982) the debut was scheduled for Autumn 1984 at the price of 499 Dollars. Later, this debut will be postponed, putting Sinclair in financial difficulties.

Some observers had a well-founded suspicion that there would be delays in delivering the QL. Meanwhile, at the end of February, 9000 bookings were recorded, rising to 13000 at the end of April. Sinclair did not hesitate to cash in the sums of money paid in advance.

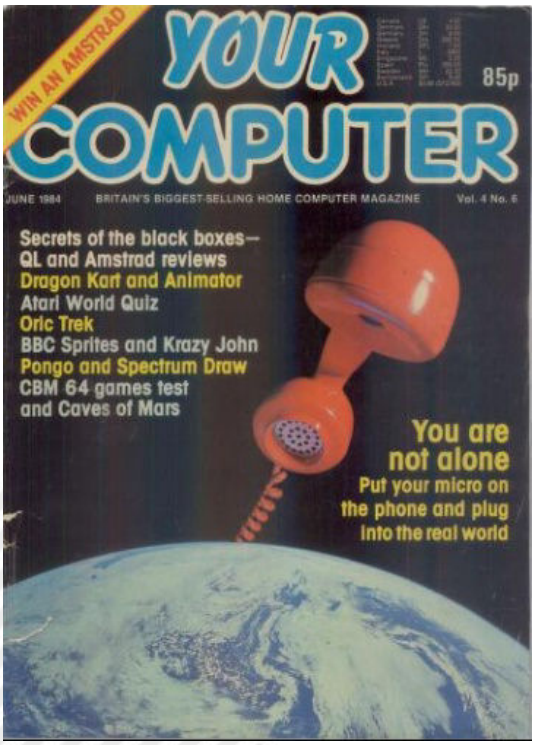
The rush to launch had created a lack of communication within Sinclair: the marketing group knew that the QL was equipped with a Basic Spectrum and that there had to be a two-phase release. The pre-release documentation for printing was scoped from the ZX Spectrum manual. But they didn't confront the developers. The result was that, having told the world that the QL was sold with a full version of the SuperBasic, the company now had to provide it.

The fact that the QL did not have the promised characteristics and that customers were exposed financially without receiving the computer at all aroused the "attention" of the Advertising Standards Authority (British equivalent of Italian A.G.CO.M., the Competition and Market Authority, n.o.s.).
Clive Sinclair promised that the money paid would not be touched and that a "trust fund" would be created until the QLs were delivered. Someone contested the fact that Sinclair, during the delay in delivery to buyers, still received interest on the sums of money paid. Then Sinclair allowed the orders to be cancelled, refunding the money. But few

seemed interested in this opportunity.
By mid-March 1984 (eight weeks after the start of bookings), no QL had been delivered and the unflattering nickname "Quite Late" was beginning to circulate. Officially because the completion of QDOS had taken longer than expected and one of the two ULAs required further modifications.
It was also to be considered that the Microdrives for QL had taken more than a year to cure various mechanical and electronic problems.

A digital circuit called P.L.L. ("Phase Lock Loop") was designed to decode the magnetic signal (and also take into account changes in the speed of the tape) but this circuit had problems recognizing the waveform of the Microdrive signal. With the help of Ben Cheese (an engineer at Sinclair who was in charge of Analogue Electronics) improvements were achieved but still insufficient. The unreliability of the Microdrives would have proved fatal to the product.

A second deadline for delivering the QL to customers was set for the end of April 1984. Finally, on April 30, 1984, it was announced that a few dozen QLs would be delivered in person by Sinclair employees to the new owners. They would get for free RS232 cable for the printer (which



Your Computer (June 1984)

cost £15).
What Sinclair's spokesman called a "goodwill gesture", angered customers who had to deal with a computer that did not mitigate the long wait endured. The June 1984 issue of "Your Computer" magazine reported that " Shoddy finish and unloadable software seems to be the least of their problems. ... The Screen Editor can make the system crash and the promised Real-Time Clock is missing - along with the manuals."

The RTC device will never be implemented because ZX8302 had to be modified to eliminate a bug regarding access to the 68008 bus that occurred during the resets. It was easier to eliminate the battery and stop saying that the QL had AN RTC.

The QL also had an external ROM (known as a dongle). According to official sources, the basic QL software (QDOS plus SuperBasic) occupied 40 KB. Too much for one 32-KB ROM. Tebby argued that QDOS and SuperBasic were already ready to be wired into ROM since March 1984 and that there was no need for an external ROM. In addition, the QL specifications provided for 64 KB OF ROM and the motherboard incorporated 2 slots for ROM chips (which could be 8, 16, 32 KB). But then how were things really going?

The Sinclair philosophy was to "deliver" and then "solve" problems rather than delay until everything was okay. In the case of QL it was perhaps a "political" move to blame the software instead of admitting that the hardware was not ready. Defective software implies tacitly that a better version is being processed while no one is willing to agree to buy defective hardware. Obviously the customers believed that naive lie and would not blink if Sinclair had actually replaced their machines in due course.

Towards the finish line for stepwise approximations

During the first six months of 1984, many other problems were solved with hardware updates and many others using software tricks.
But not all remedies led to the desired success: for example, eliminating a "glitch" that occurred on ZX8301 chip when it overheated, had minimized the impact of a bug in the use of Microdrives but had "destroyed" networking compatibility with the ZX Spectrum Interface 1.

By July 1984, the QLs no longer had the infamous external ROM and those leaving the Datatech production lines were marked "D06". At the end of the month Sinclair said that the LQs with the external ROM would soon be recalled for a free replacement of ROM. Old machines would have been demolished and replaced by "D06" or newer versions. The intention was to stagger the recall of the machines and it was not yet known how long the customers would

be without computers. The "recall" began in August and was calculated to last 10 days.

At this point a number of codified versions of ROM followed: "FB", "PM", "AH". "FB "was the oldest, "AH" was the "final" version as a spokesman said in July 1984. According to Tony Tebby, the "real" final version was the "JM" tested and delivered in March1984 and featured on QLs since July 1984 (the others were bug-resolved versions with no new additions to the SuperBasic).
A few weeks later, "TB" and then "JS" appeared (after which Tebby left Sinclair).
In 1985 there was still the "MG", which formed the basis of non-English QLs.

QL delivery to retailers began in Autumn 1984. With the new production specifications the version "D14" had been reached and the motherboard was marked "Issue6", in which an additional TTL chip healed a defect in ULA ZX8301. The hardware of the QL was stable enough to be considered "1.0". Future construction updates were made to improve things not included in the original design. The complete QL manual was written by Roy Atherton of the Bullmershe College Computer Centre, incorporating material produced by the programming staff (consisting of Steve Berry of Sinclair and Dick de Grandis-Harnson of PSION).

The high frequency oscillator of the TV output had been moved away from the left Microdrive head amplifier.
This prevented interference during the reading phase that made the drive practically useless. On the drive itself, a capacitor was placed in parallel with the head to block the disturbance induced by the drive motor. The 80s phone style connectors usedfor joystick and serial ports were replaced by 9-pin D-Sub, male for joystick ports, female for serial ports.

With QL in stores, Sinclair was lucky enough to reboot the computer. The TV ad space was packed with commercials showing that QL was a cheap alternative to its rivals: IBM PC, Mac and BBC Model B (priced from £399 to £1632 if you added a couple of floppy drives, a monitor, etc.). On the other hand, the QL proposed with the colour monitor cost £698.

But the computers were still having problems. There was still a fault with the signal generator for RGB monitor contained in ZX8301, which could have burned the chip if the monitor cable had been disconnected while the computer was still on.

In November 1984, a Sinclair User reporter claimed that out of 1000 QLs delivered to Dixons (a household appliance sales chain, n.o.s.) only 190 were working well. It was said of a shopkeeper who repeatedly tried cartridges with Psion programs on QLs to find a combination that could be read.

Despite Sinclair's insistent request for support from the most famous software houses, the shortage of software did not help sales, nor the fragility and instability that afflicted the early models released prematurely. A spokesperson for the WH Smith chain reported that sales at the end of 1984 had been very low. A spokesman for the Boots chain was talking about disappointing sales. The press estimated that only about 40,000 QL, a fraction of the potential catchment area, had been sold.
Finally, at least one of the accessories promised by Clive Sinclair (including 512 KB memory expansion, a hard disk and a modem) has yet to be seen.

Game Over

In early 1985, David Karlin intended to leave Sinclair to start his own business. Tony Tebby had kept his promise to resign only when the QL had been in a position to be distributed to customers. Jan Jones, by then expecting her first child, resigned shortly afterwards to devote herself to her family and her new career as a prominent novelist.

However, Nigel Searle managed to convince Karlin to remain in the company to manage the production of QL, assigned to two companies: Timex and Thorn EMI-Datatech and Samsung. As a result, Karlin worked on orders and invoices for 12 months, replacing Dave Chatten, appointed Managing Director in March 1985 with Bill Jeffrey (the latter from Mars Electronic). Thus Sinclair (also due to other bankruptcy projects such as the Wafer Scale Integration and the C5 tricycle) found itself in financial

Alan Sugar showing a ZX Spectrum +2

I.C.L. One-Per-Desk

trouble.

In the summer of 1985, it looked like tycoon Robert Maxwell was willing to take it over by paying £12 million. But thanks to a massive booking by Dixons, Clive Sinclair managed to overcome the crisis.
In Sinclair there were rumours of a new version of QL and the development of a 128 KB ZX Spectrum with Spanish capital. The new QL 2 management machine (codename "Enigma" or perhaps "Tyche") should have had a Gem graphical interface and supported PSION's "xChange" suite. The QL 2 project was cancelled in April 1986 with the sale of Sinclair Research to Amstrad for GBP 5 million, less than half of what was estimated in the previous summer. The new owner, Alan Sugar (who hated Psion, by the way) cut the QL in favor of PCW 8256. Many staff from Sinclair were made redundant (including David Karlin himself, who will move on to other entrepreneurial initiatives).

A total of 139,454 copies of QL were built, of which 122,793 by Thorn EMI Datatech for the British market and 16,661 by Samsung for Europe and the United States. In 1985 I.C.L. One-Per-Desk had already arrived at the price of £1195 (after a presentation in November 1984). This machine won a Recognition of Information Technology Achievement in the System Innovation of the Year competition in January 1986. A reflection of the victory, awarded to the QL in July 1985, of the Microcomputer of the Year award of the British Microcomputer Awards. Sign that someone had seen potential in the QL, but never fully realized (mainly for incorporating faulty Microdrives).

Despite the adversities, many people still wanted to continue to fight. In Stevenage, David and Vic Oliver owned Cambridge System Technology, which sold accessories for QL. Together with technician Graham Priestley, they built the Thor computer with the QL motherboard made by Samsung. The QDOS had been modified to handle one or two floppy disks. The two configurations cost £599 and £699 respectively.

Adding a 20 MB SCSI hard drive, the price went up to £1399. The specialized press immediately named him the "son" of the QL.

Amstrad obstructed its initiative by prohibiting the use of the QL name and QL components. An attempt was made to acquire the license from Amstrad but there was a clear refusal. PSION was more accommodating and licensed C.S.T., its Danish partner DanSoft and its British distributor Eidersoft for its four popular application programs.

C.S.T. succeeded in convincing Samsung to produce a compatible kit. He also managed to present the Thor 20 version in some exhibitions between 1986 and 1987, avoiding interference from Amstrad. It was even believed that it had a certain number of buyers, but in any case

The C.S.T. Thor computer

they were too few to guarantee the production that ceased in 1988.

Meanwhile Tony Tebby was designing a second generation QL with Jonathan Oakley (also former Sinclair technician). They founded QJump, a company that produced software dedicated to QL. They had also found financiers but, when it came to it, the latter werenot willing to allocate the £250,000 required by Tebby to create a finished product. However, Tebby will always remain tied to the QL world by offering accessories for QDOS and SuperBasic.

He also produced a version of QDOS called SMS2 for the

Atari ST (which had the 68000). SMS2 later became SMSQ for the Miracle System QXL, an accessory for PCs with the 68000.

Conclusion

Mistakes and unfavourable circumstances have made Sinclair QL a failure. But not all evil comes to harm. Excellent users of QL include Linus Torvalds (the inventor of Linux), who has never been mysterious about learning to program on a QL. Other companies analyzed the history events of QL and benefited from it by trying not to repeat the same mistakes.

For fun, we sometimes talk about Sinclair QL. With some contact at University of Cambridge, Clive Sinclair could get a reliable Unix operating system to customize it without wasting time. Architecture should have had a couple of floppy-disk drives instead of those horrible Microdrives.

With the Operating System and SuperBasic on diskettes, instead of being wired into ROM, fixing errors would have been easier and buyers would have endured bugs in exchange for increasingly up-to-date versions.

All you had to do was to mount a safe outlet for a monitor and then sell an external adapter and you could also use a TV. Good idea to use two RS232 ports and maybe make sure to expand the system by adding other devices to connect through the slot (for example a joystick interface or a modem). Bad idea to sell it with only 128 KB instead of immediately supplying all 640 KB. If the QL had also cost GBP 599, the enthusiasm for Sinclair was such that many people would still move from ZX Spectrum to QL.

Finally, the supreme market blow would have been to equip the QL with a ZX Spectrum emulator, capable of still using the enormous amount of games produced at the time. But, as they say, it's just talk…

Appendix

In Italy the Sinclair QL was presented on 20 February 1984 at the Hotel Michelangelo in Milan by Charles Cotton (Business Manager Sinclair) and was sold for the price of 1,300,000 Lire. He was greeted with curiosity ("MC Microcomputer" n.28, March 1984, www.issuu.com), then revealed a disappointment as evidenced by the reviews on "MC Microcomputer" n.31 (June 1984) and n.32 (July-August 1984). A similar situation can also be found in "Sinclair Computer", the "institutional" magazine of the Sinclair world in Italy (see nos. 2,3 8,9,11 and then the headings started from 14 to 19 before the merger in the magazine "Personal Computer" in January 1986).
Nowadays, looking on the Internet you will find numerous web pages, sites on the subject and events organized by fans' clubs. For example, on October 14, 2018, the 15th Italian Sinclair QL Meeting was held in Modena.

Here are some interesting links to learn more

1) www.sinclairql.it managed by Davide Santachiara,
2) www.sinclairql.net/chronology.html of Urs König (with a history of the vicissitudes of the QL),
3) www.dilwyn.me.uk of the mythical Dilwyn Jones (where there is the section "QL Emulators" from which to download the QL emulators),
4) sinclairql.speccy.org/archivo/docs/docs.html (a page in Spanish full of useful manuals in pdf format),
5) http://www.archeologiainformatica.it, an article by Stefano Paganini and Carlo Santagostino of Jan. 6th, 2014,
6) https://archive.org/details/sinclaircomputer, all issues of Sinclair Computer,
7) https://issuu.com/adpware, where you will find all the issues of MC MicroComputer magazine.

Bibliography

[AK86] I.Adamson, R. Kennedy, "The decline of Uncle Clive", New Scientist 1512 , 12 June 1986, pp.33-36.

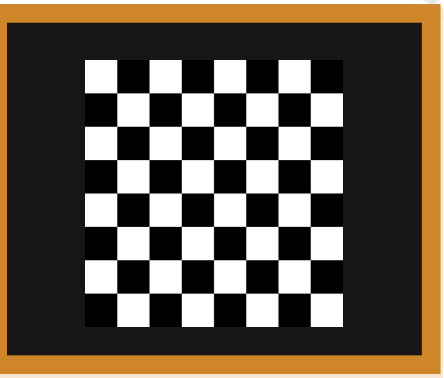
[Lea16] T. Lean, "Electronic Dreams: How 1980s Britain Learned to Love the Computer", Bloomsbury Publishing, 2016.

[NS84] AA.VV., "Sinclair's latest computer is faulty", New Scientist no. 1411, 24 May 1984, p. 5.

[Smi14] T. Smith, "Sinclair's 1984 big shot at business: The QL is 30 years old", https://www.theregister.co.uk/, accessed October 29, 2018.

[Sto18] AA.VV., "Sinclair QL: l'inizio del declino e l'arrivo di Amstrad", https://www.storiainformatica.it, consulted on October 29, 2018.

CYRUS (ZX SPECTRUM) VS. COLOSSUS (ATARI 800XL)



Man vs Computer and Computer vs Computer

Cyrus Chess vs Colossus Chess, or Richard Lang versus Martin Bryant. Who are these gentlemen?

They are two British programmers who, in the late 70s and early 80s, specialized in programming software able to play chess with enough strength to put even good ranked players into trouble. The history of many logic and electromechanical machines dedicated to the 'noble game of chess' is not recent, as it dates back to the beginning of the twentieth century.

Over time, notable names in information technology such as Von Neumann, Turing, Wiener, Zuse and Shannon have also been involved in building computers and developing specific algorithms. The challenge of beating humans at this charming game has always been an excellent application ground for mathematicians and computer scientists (see also David Levy's famous bet, who in 1968 betted that no chess computer would beat him within 10 years).

To date, the "battle" between man and machine is basically over. The machines, or rather the chess engines have prevailed over the best human players in the world for at least fifteen years. Engines developed in the traditional form have now achieved their peak and programs such as Stockfish, Houdini and Komodo easily exceed the most valued human champions by about 600 ELO points.

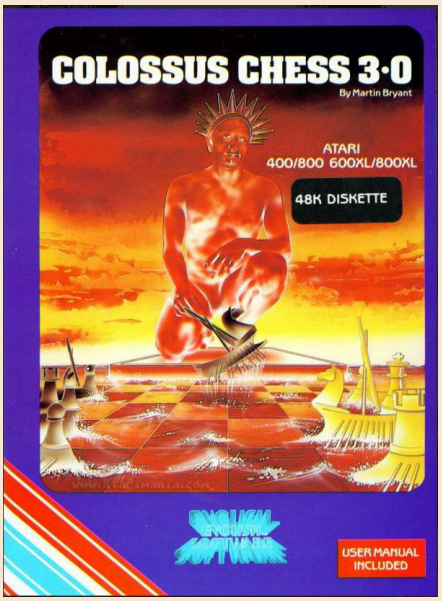
On paper it is actually an abyss. The current "human" world champion is the Norwegian Magnus Carlsen, followed closely by Italian American Fabiano Caruana (up to two years ago he used to play for the Italian Federation, then he joined the USA), the "U.S." players Nakamura and So, the Russians Mamedyarov, Karjakin, Kramnik and Svidler, the Indian Anand, etc.) and everyone of them has been using software engines for some years now to train themselves as well as to "learn" their style of gameplay.

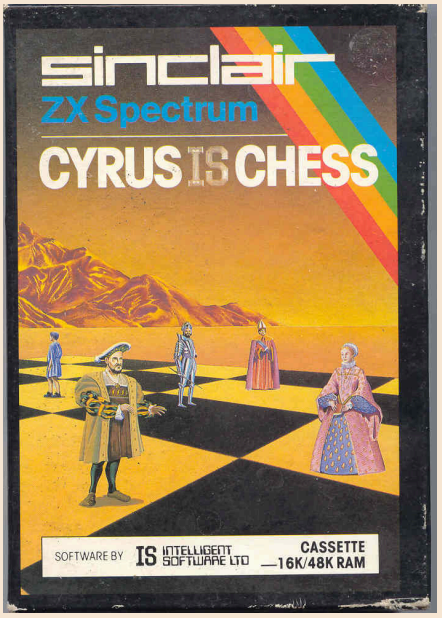
For some months now in chess software industry (and more generally in strategy games) what is happening is a real Copernican revolution: the algorithm based on Artificial Intelligence called AlphaZero, developed by DeepMind, an AI company controlled by Google, not only has been able to repeatedly beat the world champion of Go, but has brilliantly defeated the best chess engine Stockfish in a challenge out of 100 games held last December 2017 (results: +28 =72 -0, that is 28 wins, 72 draws and 0 losses).

AlphaZero, in just 4 hours of training playing against himself, has "learned" to play and then smashed Stockfish v8 with 25 victories and 25 draws with the white pieces and 3 wins and 47 draws with the black pieces. An unexpected and, to say the least, sensational achievement in the world of chess engines, which breaks new barriers in AI software applied to problems similar to complex games such as chess.

Chess and home computer: the protagonists

Let's now have a look to our





beloved retro-computers. In the early 80s, with the advent of home computers, programmers like Lang and Bryant were hired by the emerging commercial software market to develop their successful algorithms on 8-bit machines. We can appreciate their results in two chess programs that we have selected for a match/review on these pages. Yes, we have made a little research to determine which of the two algorithms was the strongest at the time, comparing two of their best implementations: Cyrus Chess on Sinclair ZX Spectrum and Colossus Chess 3.0 on Atari 800XL.

Richard Lang began his career as a specialist programmer in January 1981 after thoroughly studying a book of Dan and Kathe Sprackler about one of the first algorithm named Sargon. That book even provided a source code in Assembly Z80. So Lang worked hard and found new ways to improve the theoretical scheme Sargon was based on, not only to make it faster but also to implement more advanced techniques and a more efficient system to evaluate positions reached on the chessboard.

His first release, Cyrus, won the second European Chess Championship for Microcomputer held in London in September 1981, with 5 wins out of 5 games played. Lang was immediately offered a contract by Intelligent Software (which was founded by David Levy) and Cyrus Chess for ZX Spectrum was his first commercial title. Lang later moved on to port Cyrus to several platforms on behalf of Intelligent Software and in 1983 he moved on to the new Psion program for the 68000 processors, whose first release was the version for Sinclair QL.

In 1985 Lang collaborated in porting the Psion engine to the famous Mephisto series of chessboard computers that, together with its software version called Chess Genius, dominated the scene of dedicated microcomputers from the mid-80s until the early 90s, winning several times the title of World Champion in its category. In 1994, Chess Genius even won a quick game (25 minutes for each side) against the then world champion Garry Kasparov. Since 2002, Lang has been managing his small company that sells Chess Genius for various platforms, PDAs and smartphones, including the Android version. Lang recently stated that even today many of his original ideas and techniques used in Cyrus still live in his current releases.

Martin Bryant began experimenting with his first chess program in 1976, released under the name of White Knight, which was initially developed in Pascal and then brought to Assembly 6502 for the Apple II. His early version of the engine won the European Chess Championship for Microcomputers in 1983, a year later after Cyrus.

White Knight was released in two versions for BBC Micro and Acorn Electron and its main feature was to calculate and display the strongest variant in a given position, the so-called "Best Line" function, which later became a common feature in all chess programs.

White Knight's algorithm was used by Bryant as a basis to develop the whole Colossus Chess series since 1983, with titles and versions for a large number of 8/16-bit platforms of the 80s, including Amiga, Atari ST and IBM PC. In 1985 the magazine Zzap!64 awarded Colossus Chess the prize of “best chess program” for the home computers. Later Bryant never stopped improving his engine and his commercial releases under the name of Colossus Chess. The latest versions date back to 2008 when "Colossus 2008b" was released.

The challenge

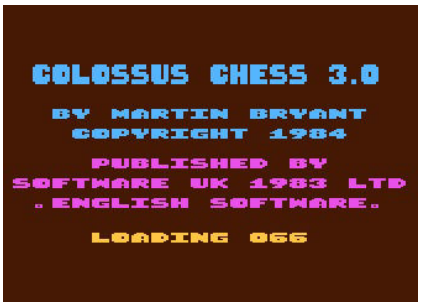
Cyrus and Colossus are both easy-to-use chess software with quite intuitive user interfaces. To start playing against the computer with White you just make the first move or access the advanced commands

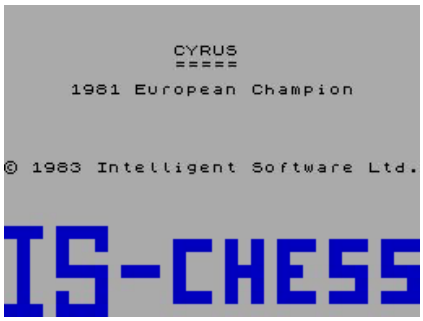
OUR FINAL SCORE

Cyrus IS Chess - Intelligent Software Ltd / Richard Lang - 1983 - ZX Spectrum 16/48K

Gameplay 75% - The user interface provides simple usage of the keyboard and the graphical representation of the chessboard is good enough to play only with the help of the screen. The game is reproduced in all its rules (en-passant, draw for repetition of moves, rule of 50 moves, playing finals, etc.). With several game levels available Cyrus is a valid opponent still today (the playing strength is estimated at around 1650-1750 ELO points).

Longevity 90% - For chess enthusiasts there is no end to the number of good level games that can be played on your ZX Spectrum. The engine effortlessly tackles all the most played openings and variants.





to change the level of play, play with Black, rotate the chessboard, set up a different position, customize colours and so on. To that respect, Cyrus on ZX Spectrum is more intuitive than Colossus as it always displays the game options. These commands can be accessed by pressing alphabet keys, whereas Colossus needs to press a CTRL-<key> combination to activate the game features. Both programs feature a graphical display of the chessboard that lets the user play a whole game using the screen only (at that time many players held a real chessboard next to the computer in order to replicate the played moves).

The match between the two programs was played by setting an equal game level for both of them in terms of play strength, with the same number of seconds allowed per move.

The emulators used to run both chess programs (Spectaculator and Altirra under Windows) have been set to recreate the stock conditions of the two home computers: a 48K ZX Spectrum 48K and an Atari 800XL.

The challenge was set in 6 games at a constant rate (30 seconds per move) and each software played 3 times with the white pieces and 3 times with the black pieces. Here is the outcome of the clash between these two historic chess programs: Colossus 4.5 - Cyrus 1.5 (4 wins for Colossus, 1 for Cyrus, 1 draw).

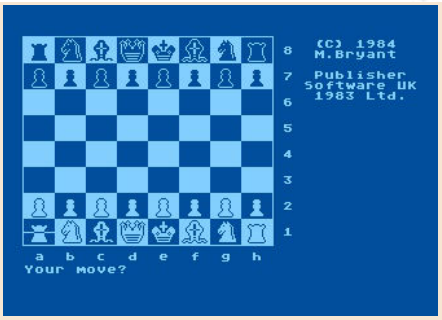
Colossus proved to be much more robust and held a good but solid style throughout the match. Cyrus got off to a good start by winning the first challenge with the white pieces but lost the next 3 matches, yet showing some original ideas when attacking its opponent and even using pure strategical tricks to stay even during the games.

Both engines diverted quite early from the canonical lines of the openings, but in the middle game Colossus' conservative technique prevailed at last. The games averagely lasted for about 40-50 moves and the only draw came for repetition of moves.

As the engines don't feature the option to abandon a game in case of manifest inferiority of material or position, the win or the draw have been manually declared (the software which was winning would have however come to checkmate the opponent). All 6 matches of the match are available on request in PGN format.

It was fun to play and test the strenght of the programs selected for this challenge. And it's amazing to see how these little home computers, with their reduced memories, can still cope with many skilled human players, such as myself!

by **David La Monaca**

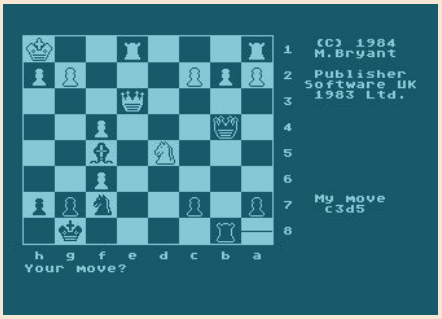


OUR FINAL SCORE

Colossus Chess 3.0 - English Software Co. UK - 1984 - Atari 400/800 XL/XE

Gameplay: 80% - The program offers a wide range of game levels, customizable to create a large number of combinations. An essential but effective representation of the chessboard allows you to check out the moves and the reached position at a glance. If you want to access the many commands available, you need to take a look at the manual. All rules of play are respected and the playing strength is around 1750-1850 ELO points.

Longevity: 90% - The same ratings expressed for Cyrus Chess are valid for Colossus. This engine was and still is a good opponent for amateur and ranked players. Surely one of the best, if not the best, chess engine for Atari and all the 8-bit machines of the 80s, very adaptable and with a high level of customization.



HIBERNATED 1

Developer: Stefan Vogt
Publisher: Relics
Year: 2018
Platform: Amiga/C64
Genre: Adventure

Almost a year ago, in the Italian version of RetroMagazine, we published a in-depth insight into this game by Stefan Vogt and Pond Software. Back then the choice was made to “only” review the unboxing of the physical edition, since the game is a text-adventure and, as such, it is kind of difficult to review visually. That physical edition was just way too good to go unnoticed since it basically contains formats and media for anything in the late 80s/early 90s: tapes (yes, tapes!), 5.25" and 3.5" floppies, and even a Micro SD card, just in case you were out of physical disk drives to try the game on and you were in need of using an emulator.

As Tony wrote back then, it is sure difficult to write about a text adventure, but I recently finished this game and thus I simply had to report back. Actually, in the last couple of years, I did write about text-adventures, namely Zork and Leather Goddesses of Phobos, but this is the first time that I review a C64 game (meaning that I played it on a real C64, but that was just my personal choice).

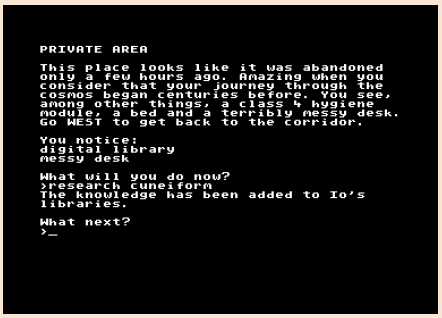
I think that the best choice here is to let the author speak, so here are his words: "Have you ever dreamed about a journey far beyond the known regions of the universe? Hibernated 1: This Place is Death is a science-fiction text adventure for C64, ZX Spectrum, Amstrad CPC, Amiga, Atari ST, and PC DOS. It is the first interactive story of an epic trilogy centered around Olivia Lund, who has been sent on an interplanetary exploration mission by the Terran Alliance. After being in hypersleep for more than 200 years and with more than 800 light-years being traveled, her ship, the Polaris-7, crosses paths with a gigantic alien vessel and is captured by a tractor beam. Olivia soon finds out that this may not be her only problem. There is no communication and there are no signs of life.

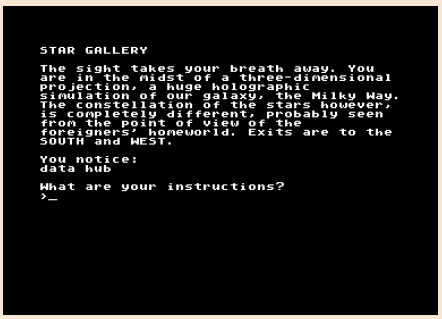
The extra-terrestrial spacecraft just keeps on drifting through the cosmic void, which is something it seems to be doing for thousands of years now. This is a tomb in-between the stars, which Olivia has to enter to extricate herself from this interstellar trap. Io, the navigation robot of the Polaris-7, is probably her only friend. Far away from home and surrounded by death and decay, she found the answer to one of the greatest questions of mankind. Are we alone? The answer is: yes, out here, we are more alone than ever."

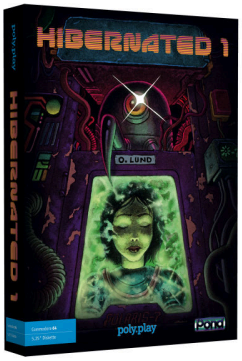
The adventure dates back to 2018 and, one year later, Pond Software published an add-on for the game (named Eight Feet Under) unveiling further details on the story. The game is available on itch.io but, despite it being a stand-alone product, it is strongly advised to play the main adventure first, to avoid spoilers. For the same reason, I won’t delve further into the story, but let me just say that a sequel has recently been announced. I really can’t wait to play it. What about you? Just be patient, and keep the flame alive.

by **Gianluca Girelli**

RetroMagazine has established a strong collaboration with **AmigaGuru**:
https://blog.amigaguru.com/a-look-at-hibernated-1-all-in-one-edition/
https://blog.amigaguru.com/hibernated1-review/







CIVILIZATION

Publisher: Microprose Software
Year: 1991
Platform: MS DOS
Genre: Strategy

"Civilization is a game that the word 'GREAT' could only adapt to if it were written in 100-meter-high cubital letters on a flashing neon on the top of a 50-storey skyscraper." With these words, the editorial staff of K Magazine Italy begins to talk about Sid Meier's Civilization in issue 34 of December 1991. And reading these words brings back to my mind the impression I had when I started playing this cornerstone of video games: immensity.

1991 was certainly not a greedy year of masterpieces, although to be honest in a crowded field such as the video games market, probably no year after 1978 was. Lemmings, Street Fighter II, Sonic, Monkey Island 2, Another World, all these titles have marked a change of pace in their respective genres, both on the technical front and playability.

MicroProse, in September 1991, emerged with the last effort of Sid Meier. The Canadian programmer had developed his project together with Bruce Shelley, then initiator of the Age of Empires saga. Meier had already released 22 titles at the time, and had made himself known especially to F-19 Stealth Fighter and Railroad Tycoon.

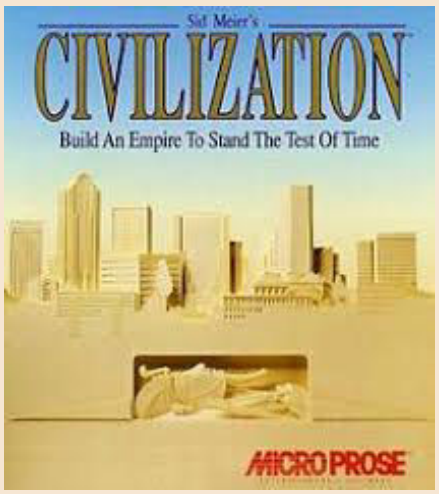
There is no doubt that Civilization is a turning point in the world of video games, on many fronts, and in fact one of its main characteristics is precisely this faceted nature. Civilization is a wargame, but it lacks a specific goal, if we exclude victory by "universal conquest". It is a diplomatic simulation, but it is intertwined with the military aspect as the enemies react to the presence of our troops on their soil. It is also a race for technological advancement and scientific progress, but they depend heavily on how we manage cities, as they provide 'brains' to our schools and universities. It is even a management game, since the resources of cities must be managed with care to avoid problems of famine or production stagnation.

It goes without saying that the detail with which the individual aspects are addressed is less than what dedicated simulations have (or will have), such as Sim City, but the amount of factors to take into account is really high and despite this, the game remains extremely engaging and fun. This is certainly what makes Civilization not only a vast game, but a 'great' game, where the apparent lack of actions of the initial settlers turns into a universe of possibilities, still remaining within the limits of the game and avoiding to turn into a brain puzzle.

We will find ourselves starting the game with a caravan of settlers, after witnessing the "cosmic" presentation and choosing the civilization that we want to impersonate, the level of difficulty and the geography of the world. The game does not yet include the concept of 'fog of war', limiting itself to completely blackening the parts of the map that we have not visited. At the beginning of the game, therefore, we have visibility only on the 8 tiles surrounding the settlers. If we are lucky enough, the game will assign our tribe a free technological advance (e.g.

Technical requirements

The graphics and sound sector are certainly not the strong point of Civilization, an aspect that the subsequent chapters of the series will take better care of, but there is no great lack of it. The interface is clean and tidy and the numerous shortcuts that the units provide allow you to do most of the work without having to navigate between menus or use the mouse. This device is still needed to fully appreciate the game, but this recommendation is not needed in 2018.



alphabet or ceramic) and/or an additional unit, which in the early stages of exploration and defense can definitely make a difference.
The game then continues in turns when we can move all the free units, but there is no rush in Civilization. At the end of the shift we have all the time to visit the cities that we intend to found and to monitor their progress, to plan production, to strengthen their defences. And we will need to take this time, as soon as we begin contacting the neighbouring civilizations, sometimes eager for friendly relations, sometimes only eager to see our empire crushed beneath their primitive feet.
Relations with other empires will be a constant factor in the game and a considerable thorn on your side at certain crucial moments. There is always a foreign city already built in the area where we would like to found our new capital, there is always an enemy unit in a strip of land that leads to unexplored regions, preventing our units from failing to declare war, there is always an opposing nation that discovers the atomic bomb before us or that starts to build spaceships when we have just discovered the gunpowder. You don't rest on laurels in Civilization, and if you do, you better get ready to smell the stench of death, because someone who wants to set our glory bed on fire is always there.
Unless, by mastering the military, economic, scientific, political and social factors of the game with incredible skill, we come to build a truly powerful empire, but you will quickly realize how much difficult it is to manage all these things at the same time. Even the most powerful empire, when attacked wisely, can begin to limp. The loss of a city, for example, means the loss of military production, wealth and scientific contribution, but also the disruption of the lines of communication of our empire, or perhaps the longed-for access to the ocean.
Exploration is an important theme in Civilization, as finding the right place to found a city can make a big difference, especially if you manage to exploit special land, which therefore produces more than the average, especially if properly treated by our trusted settlers. Exploration, however, is linked, as is the military power, to scientific progress. It is not enough to invent the trireme ships to be able to sail the ocean, so that these units must end the shift next to the coast or they will be lost at sea. It will be necessary to discover navigation in order to venture into the midst of the water regions that separate us from lands where our nation can thrive. Or at least this is our hope when we load settlers and a couple of military units onto our walnut shell, since at first we don't know what's beyond the water, and WE don't even know WHERE this phantom destination is. We don't even know about its very existence. This would happen, of course, unless we play on a known map, but also there, as our initial position and that of our opponents is random, we could happily land in America only to find out that the glorious Russian empire has already decorated the continent with cities named very differently from those that the Pilgrim Fathers have taken from their motherland.
Technological advancement is therefore an important element of the building that we will have to raise by accepting the challenge of Civilization. The tree of scientific discovery is definitely dense and, if at the beginning of the game the choices may seem small, once again it will take little time to face typical dilemmas between the development of something that allows us to improve the conditions of the population, such as a new form of government or a productive process, and the new military technology that will finally put all enemies under our feet. On the other hand, having prosperous but unprotected cities is as useless as dominating the world and having the population at the end of its rope.
Planning in time where you want to get to is therefore essential, so you will need to consult often the descriptions of scientific discoveries and units. And speaking of descriptions, you can't help but mention the huge online

Bugs

Like any software, especially of this size, Civilization is not bug-free, but the overall care of the product is excellent, so much so that errors can only be found in particular sequences of game events. A very famous mistake that is easy to come across is what sees the peaceful Indian leader Gandhi change his attitude dramatically from friendly to relentless warmonger. The origin of this bug lies in the (simple) aggression management system of a leader, which is represented by an integer between 0 (absolute peace) and 255 (total war), evidently an 8-bit integer. Specific game events reduce the aggressiveness of a nation's government, and specifically the transition to democracy reduces the aggressiveness of 2. The programmers, however, forgot to check the lower threshold of this value and Gandhi goes from 1 to -1, a negative number that turns into 255 (since an integer hosts only positive numbers), transforming the Indian nation into a den of bloody assassins who only want to see our cities razed to the ground and our population passed by the sword. When you say it takes one nothing to start a war...

manual. The documentation available to the player is not only well done, but covers every part of the game, and it also provides details of every unit, city infrastructure or scientific discovery that can be accessed. Today with Wikipedia and Google we are all used to being able to access knowledge that exceeds by many orders of magnitude what we imagined 30 years ago, but in 1991 "Civilopedia" was a monumental work, which contributed significantly to that impression of immensity that you feel playing Civilization.

Online Game

Nowadays it is impossible to think of a game like Civilization without immediately imagining the clutches of players facing each other online, but in 1991 this possibility was not contemplated, also because it would have objectively found the favor of a few lucky players possessing a modem and the budget necessary to access the telecommunications networks that we can safely call primitive compared to what we have now. Four years later the landscape of telematic access had changed and MicroProse released CivNet, a version of Civilization that allowed 8 players to face each other online or on the same computer. This version of the game, however, came out too close to Civilization II, published the following year, and spread little.

So you will discover the many buildings that can be raised in our cities, each with specific advantages for productivity or social needs. Among these we must mention the 21 wonders of the world, which, once built in a city, bring a global advantage to our entire empire. These range from the Gardens of Babylon, which make happy citizens in every city, to the Apollo Program that allows space travel and reveals the entire map. It is very depressing when, two turns after the end of the construction of the long-awaited Pyramids, one of the opposing leaders announces to the world that they have just finished them, forcing us to change running plans, usually throwing away shifts spent on their production. Military units are also very diverse and benefit from scientific discoveries. Discovering iron working allows us to start building bridges but also to produce the dreaded Legions, and properly exploiting the latest discovery of our scientists can be the key to the final victory.
Civilization also allows for quite paradoxical situations, when one empire finds itself to be remarkably advanced if compared to another. This rarely reaches extreme levels, as weak and underdeveloped empires are quickly eaten by the most powerful neighbors, but sometimes you can see wooden catapults facing tanks and when it happens the combat system creaks a little. Units in fact have attack, defense and movement parameters that are comparable to other units of the same period but not to units of different historical periods. A catapult has an attack value of 6, while an artillery has a defense of 2. This basically means that you can get rid of a robotic vehicle armed with ground-to-ground missiles by throwing stones with a wooden arm. Or a group of medieval knights could hold a modern city under siege! Anyway, as already mentioned, these elements are marginal and do not affect the gaming experience. It will be difficult for your settlers to face a German bomber-fighter aircraft, as Frederick the Great has already conquered you as soon as he discovers the gunpowder. For 5 years, until MicroProse published the beautiful sequel, Civilization put the sceptre of total command in our hands, made us dream of creating empires "where the sun never sets" and reaching the stars with our spaceships, ready to plant our flag on a new planet. Thank you Sid and Bruce, you have given us years of absolute fun, delicate social and military decisions, absolute and perceptible "immensity"!

by **Leonardo Giordani**

A large family

In 2018 Firaxis released Civilization VI, a testimony that the initial idea was brilliant enough to last for 30 years in the rich video game landscape. Obviously we are now faced with a product that is completely different from a graphical and sound point of view, and the algorithm behind the game has also changed, especially as regards the intelligence of the opponents. These modern incarnations, however, despite the obvious visual adaptation and the possibility of exploiting the power of today's computers, do not deviate in a particular way from the spirit of the forefather, who, with the poverty of means of time, had already given players all the means to fulfill their dreams of greatness.

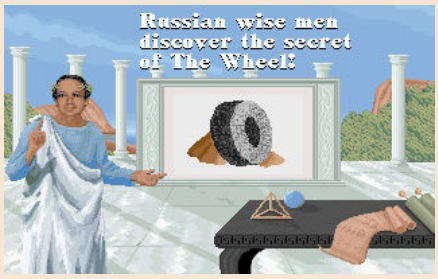
OUR FINAL SCORE

» Gameplay 90%

Civilization is a turning point in the world of video games, not only in the strategic ones.

» Longevity 99%

Thank you Sid and Bruce, you have given us years of absolute fun, delicate social and military decisions, absolute and perceptible "immensity"!



THE PAWN

Publisher: Ranbird
Developer: Magnetic Scrolls
Year: 1985
Platform: All platforms
Genre: Adventure

In our journey about re-discovering textual adventures, whose episodes in RetroMagazine are randomly placed in time, it's the moment to talk about Magnetic Scrolls and their fantastic games. In this article we will briefly tell the history of this excellent software house and review The Pawn, the adventure game that announced the whole world the existence of a worthy rival of Infocom.

Magnetic Scrolls: the software house

In the middle of the 80s, more precisely in 1984, Anita Sinclair, Ken Gordon and Hugh Steers decided that the time had come to challenge Infocom (born in 1979) creating, in London, a software house for the production of text adventures that could compete with those of the American company both in story-telling, by developing new stories with twisted and engaging plots, and technically, thanks to an extremely advanced parser and a design set to give the player as much freedom as possible, thus expanding the user experience beyond the mere goal of the game. The company started with a team of eight permanent employees and a variable number of freelance collaborators, with Sinclair as general manager, Gordon as technical director and Steers, who had the greatest programming skills, as chief programmer. The plan was to produce games mainly for the Sinclair QL, a microcomputer owned by Anita, but the release of the Atari ST and the Commodore Amiga, with their hardware and OS superiority, convinced the three founding partners to expand their initial objectives and increase development efforts to support these two platforms as soon as possible. The result of their work was The Pawn, the first game produced by Magnetic Scrolls, developed by the 18-year-old Rob Steggles, who later became the author of the four most appreciated adventures of the company. Like Infocom and in general like most of the software houses involved in the production of text adventure games, the guys at Magnetic Scrolls chose the path of creating an interpreter (or virtual machine if you like, since that's what it was) allowing the execution of games on multiple hardware platforms without the need for a total rewrite of the game. That would prevent, or at least minimize, the burden of porting the whole code on different platforms, which would have been too expensive and time-consuming. The entire gaming system (today we would say game engine) was created by Steers and Gordon and it had one of the best parsers ever seen, pretty comparable to the one developed by Infocom. In addition to that, graphics and sounds could be included to the stories, although these additional features were not available for all platforms. In early versions, programmers compiled the game source files with the help of a normal text editor, but later there was a graphical tool available, produced thanks to the joint work of the Sinclair/Gordon/Steers trio, to speed up the production process and allow developers to focus more on game design than writing code. During its period of business activity, Magnetic Scrolls developed 8 games (see box). All these adventure games featured a level of difficulty higher than average, an excellent parser with a very wide vocabulary and very high quality graphic illustrations. The brilliant career of Magnetic Scrolls ended when the public interest shifted to other genres of games, graphic adventures in the first place. The production of text adventures stopped generating profits for the survival of the company and so the brilliant partners were forced to shut down the business. The games created by Magnetic Scrolls during their time of production are, however, still appreciated and played today. Moreover they are considered among the best ever.

Magnetic Scrolls' games

The Pawn, 1985, published by Rainbird for Amiga, Amstrad CPC, Amstrad PCW, Apple II, Acorn Archimedes, Atari ST, Atari 8-bit, Commodore 64, MS-DOS, Macintosh, Sinclair QL and ZX Spectrum

The Guild of Thieves, 1987, published by Rainbird for Amiga, Amstrad CPC, Amstrad PCW, Apple II, Acorn Archimedes, Atari ST, Atari 8-bit, Commodore 64, MS-DOS, Macintosh and ZX Spectrum

Jinxter, 1987, published by Rainbird for Acorn Archimedes, Amiga, Amstrad CPC, Amstrad PCW, Apple II, Macintosh, Atari 8-bit, Atari ST, Commodore 64, MS-DOS, ZX Spectrum

Corruption, 1988, published by Rainbird for Amiga, Amstrad CPC, Amstrad PCW, Apple II, Acorn Archimedes, Atari ST, Commodore 64, Apple Macintosh, MS-DOS, ZX Spectrum

Fish!, 1988, published by Rainbird for Amiga, Amstrad PCW, Apple II, Acorn Archimedes, Atari ST, Commodore 64, Macintosh, MS-DOS, ZX Spectrum

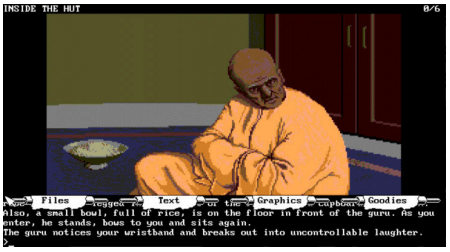
Myth, 1989, published by Rainbird Amiga, Amstrad PCW, Atari ST, Commodore 64, MS-DOS, ZX Spectrum

Wonderland, 1990, published by Virgin Interactive for MS-DOS, Acorn Archimedes, Amiga, Atari ST

The Legacy: Realm of Terror, 1993, published by MicroProse for MS-DOS systems only. There was also a version for Amiga that was never completed.

The Pawn

"You wake up on a sunny August morning with birds singing, and the air fresh and clear. However, your joints are stiff and you have not woken up in your bedrooom as you wold expected. Trying to recall what happened the night before, you manage to piece together a few brief glimpses to give the following account: You were walking home, having just done your week's shopping at the supermarket, very close you noticed a stranger in a white overcoat coming toward you. When he got very close you noticed that he was wearing glasses and had a thick, bushy beard. As he



passed you he left out a hollow cackling laugh and you felt a sharp blow on the back of your head. Then you woke up. You notice that you are wearing a silver wristband which covers your forearm."

This is how The Pawn, the first and most beloved Magnetic Scrolls game, begins (the text is a summary of the opening paragraph of the game). Designed and programmed by Rob Steggles, with the support of Steers and Gordon, during his summer vacation (Steegles was only 18 years old at the time), this first work by Magnetic Scrolls is appreciated for its fun story, well constructed and told. It is also remembered for the use of stunning (at the time) graphics screens appearing during the game and for a high-quality parser, easily able to compete with the one created by Infocom. The wise use of images in text adventures was subtly introduced by smaller software houses, trying to attract more players. They were supposed to be seduced by illustrations and perhaps overwhelmed by the quality of the plot and the parser, as most of these games featured a horribly low parser, sometimes quite irritant and working unexpectedly. Both the Infocom guys, whose motto was "a paragraph is worth a thousand images", and those at Magnetic Scrolls didn't like to use graphics in their games, convinced that well-written text and a robust plot would be more than enough to stimulate players and get them into the atmosphere of the game. In order to release The Pawn, however, the publishing label, Rainbird, demanded the adoption of pictures within the game under penalty of not agreeing on the distribution. Although strongly upset by this ultimatum, the directors of Magnetic Scrolls asked Geoff Quilley to develop some pictures to go with the plot and when the dynamic trio saw the sketches, they totally changed their minds about the use of the images in their adventure games. Quilley presented incredible quality drawings, magnificently rendered on the powerful Atari ST and Amiga. The decision was soon made: instead of using pictures only to faithfully show the game scenes, a task whish until then was left to the text and the imagination of the players, they were adopted to enrich and enhance the final product, more or less like the illustrations were used in the great classic books. They now were a pleasant amusement for the reader/adventurer, giving him a few moments of leisure from the commitment required by the adventure. Quilley's mastery later allowed the graphics screens to be successfully adopted even on 8-bit microcomputers the games were released for. This was not the only touch of class that distinguished The Pawn and all other Magnetic Scrolls games. A well-groomed package, an introductory novel written by Sinclair, an elegant use of humour (British humour, of course) and an original help system contributed to the wide success of the games. And some of them even took a well-deserved place in the Olympus of the most beautiful textual adventures of all time (The Pawn was awarded with prizes like the Golden Joystick Awards in 1986 for being the best adventure game of the year. Wandering around the locations that make up the discreetly extended world of play and reading the novel in the box (which we strongly recommend you do before you start playing), you will discover that you have ended up in Kerovnia, a fantastic land in an unspecified time and ruled by King Erik, in a crucial moment of its history due to social riots. Soon you will realize that fulfilling your desire to return home is not an easy thing to do. In The Pawn, danger is everywhere and, although death will not wait for you around every corner, the chances of ending up in another world (no, not your home!) are high enough to advise you to save the game every time you have a chance to. Moving the first steps will not be difficult, either because the size of the game map will give you a certain degree of freedom before you encounter an enigma to solve in order to access a new area, or because the first riddles you may encounter will not be so hard to overcome. In the early stages of the game, after about half an hour, you will have already met the magician shown on the front page of the game box and the King of Kerovnia, but from now on things will start to get much more complex. Although puzzles always have a logical solution, solving them will not be at all trivial: you will have to squeeze your meninges and take advantage of the creative use of the game parser, sometimes a little pedantic. For example (SPOILER ALERT!) in the search for a way to get rid of the bracelet, when you meet the magician intent on flying on a prairie, after greeting him and accepting the assignment that he offers you, try to "show the wristband to Kronos". The magician will tell you that it's nice. But if you "ask Kronos about the wristband", the answer will be completely different and definitely more helpful for your purposes (or so it will seem). Why set

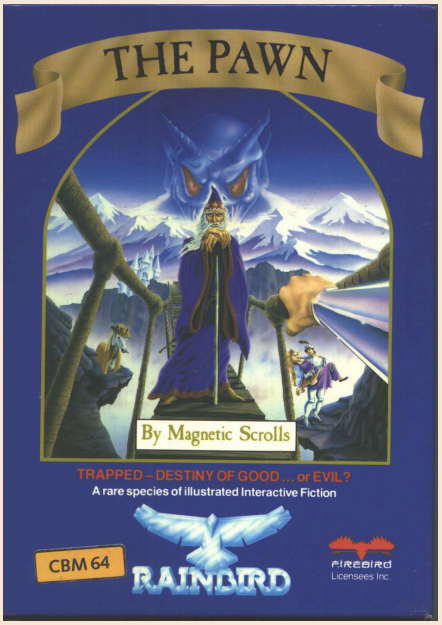
such a trap to the player? It is clear that if you show the bracelet to someone, that is to receive useful information about it, or to understand why you are wearing it and how to remove it. What is the reason to complicate things by selecting the way you talk about it with other characters? Of course a nice exercise of style, useful to show how smart the parser is, but honestly an unnecessary complication to the detriment of the player in our opinion. You will find more than one of these situations in this adventure game, so do not give up if your typed commands do not achieve the desired result. If you have an intuition that you consider highly plausible, try to explain to the parser what you intend to do by reformulating the sentences, it might

work. Beyond these little "youth flaws" (it's a debut game, after all), the adventure is absolutely enjoyable and engaging. As you move forward in the story, it will become very clear why it has been named with such a title ("The Pawn"). Throughout the game you will

feel manipulated for unknown purposes, which will become a certainty as events unfold. The number of locations, objects and people as well as the variety of possible actions will allow you a high degree of freedom within the world of this fantasy place, and this will provide you with the opportunity to discover the many humorous tricks which the author has permeated the adventure with. For experienced players, who like to immerse themselves in the universe created by the programmer rather than just making the right choices to go on with the story to its natural epilogue, this is a great value; on the other hand, less skilled adventurers could be disoriented by such a range of possibilities and have a lot of difficulty reaching the end of the game. There is generally great attention to detail in the game, which can lead to different consequences depending on what you decide to do and how you explained it to the game parser. The way you talk about the bracelet to Kronos is one example, but it's not the only case. If for some reason (which we do not wish to investigate at all) you think about riding around in an adamythic outfit, you can do so by asking your alter ego to take off your clothes and continue the adventure in your underwear (at least guessing you wear them). No one in the game will be surprised by your clothing tastes. If, unfortunately, you forget that you are walking around dressed only in your bare skin and try to cross the snow-capped mountains, death by freezing will soon catch you, unless you get dressed quickly. This kind of attention by the game to how you approach every situation and how you relate to the other characters, magnificently expands the range of actions you can try and the different ways to solve the puzzles, thus increasing the depth and longevity of the game. And this is an element not easy to develop in a textual adventure.

Conclusions

The Pawn is a classic of the interactive fiction and one of the best works of his genre. Although it is not a title suitable for beginners or to those approaching text adventures for the first time, it deserves to be played by everyone, even overcoming the linguistic difficulty (it is only available in English), a feature that may not even be considered as a flaw. Of course, if you are not familiar with English and its nuances, the overall difficulty of the game will increase, but on the other hand this can be an excellent stimulus to increase your level of English, both in vocabulary and grammar. Many players have learned or improved their English thanks to the IF (and so do I), driven by the desire to play and have fun with this genre. Language skills aside, there are many features that a newcomer, especially if under 30, can discover and appreciate. Firstly, the difficulty of the game, linked to the intellectual exercise that adventure requires, not used artificially, so much to extend its longevity, but well structured and an integral part of the experience and development of the plot. The wide range of possibilities offered by the plot and the capacity to deeply involve the player into the adventure, even though basically it doesn't feature advanced graphics, audio, video and action, can surprise those who are more accustomed to seeing spectacular animations or facing a game counting on their manual ability rather than using their brains and lateral thinking. The Pawn's ability to entertain and engage players even today, after more than 30 years from its release, is an excellent example of the universality of video gaming and this should already convince you to give it a try. Well, if my whole praise talking had convinced some of you to try this title, we are happy to inform you that the game is still on sale, along with many other Magnetic Scrolls titles. It is also available for mobile platforms such as iOS and Android and it is even possible to play it (legally) online thanks to an interpreter written in javascript: it only takes a modern web browser pointed to: https://msmemorial.if-legends.org/msa2/msa2.html
Happy adventure to you all!
by **Giorgio Balestrieri**



OUR FINAL SCORE

» Gameplay 95%

A keyboard and your imagination is all you need to play. The Pawn is both fun, challenging, engaging and exciting. Once you start playing, it will be hard to stop. You will be drawn by the desire to see how far it may bring you.

» Longevity N/A

It's a text adventure, and you know how we usually think about the longevity of such a work. The Pawn, however, somehow features the ability to be different every time you start it again. You can be sure that something new will come out of the screen. Finding out all the details of the game will take you a lot longer than it takes to finish it (which is already a lot).

Gens Una Sumus or...
let's have some fun with our common passion!

We have come to the end of this zeroth issue of the English edition of RetroMagazine. We hope you enjoyed it and that our efforts to reach the many retrocomputing and retrogaming fans around the world have not been in vain. Our magazine in its Italian version is now three years old, but the many requests received from all over the planet for an English version convinced us to put together this special issue.

Our strengths can be summed up in three key words: fun, passion and inclusiveness.

We always have a lot of fun doing all the activities needed to prepare for each issue. We follow the best of current retro-events as well as enjoying the nostalgia factor of retrocomputing, with articles, reviews, interviews and special columns on a variety of subjects, old and new. What drives us and gives us the energy to work on the magazine is the passion for our old beloved home computers and consoles, for the games we loved as kids and for the long hours we spent learning how to code.

Last but not least this is a multi-platform magazine and we like to stay inclusive and keep all the doors open for external contributions. Back in the '80s and the '90s we used to split into as many hostile factions as there were 8- and 16-bit incarnations of the computers and consoles that each of us had bought and loved. Harsh rivalry was the order of the day. Today we may have grown up but still some of these rivalries remain. Now we believe that the fun and the passion for our machines are (please forgive the loaded term) “contagious” and that everyone can and should enjoy them.

And that's why we have borrowed the motto of the International Chess Federation (“Gens Una Sumus” – ***we are one people***). It means that we'd like to address to you, new readers of RetroMagazine, an invitation to collaborate. We want to make RetroMagazine English an effective meeting point for all retrocomputing and retrogaming enthusiasts as well as an increasingly interesting and exciting read!

So we look forward for your comments and, if you're interested, your concrete contributions. And don't hesitate to get in touch with us and propose your ideas, articles and game reviews.

David La Monaca (Cercamon)

Our contacts are:
Email - retromagazine.redazione@gmail.com
Facebook - https://www.facebook.com/RetroMagazine-2005584959715273/

Disclaimer

RetroMagazine
Year 1 - Issue 0

Chief Editor
Francesco Fiorentini

Managing Editor
David La Monaca (Cercamon)

MAY 2020